

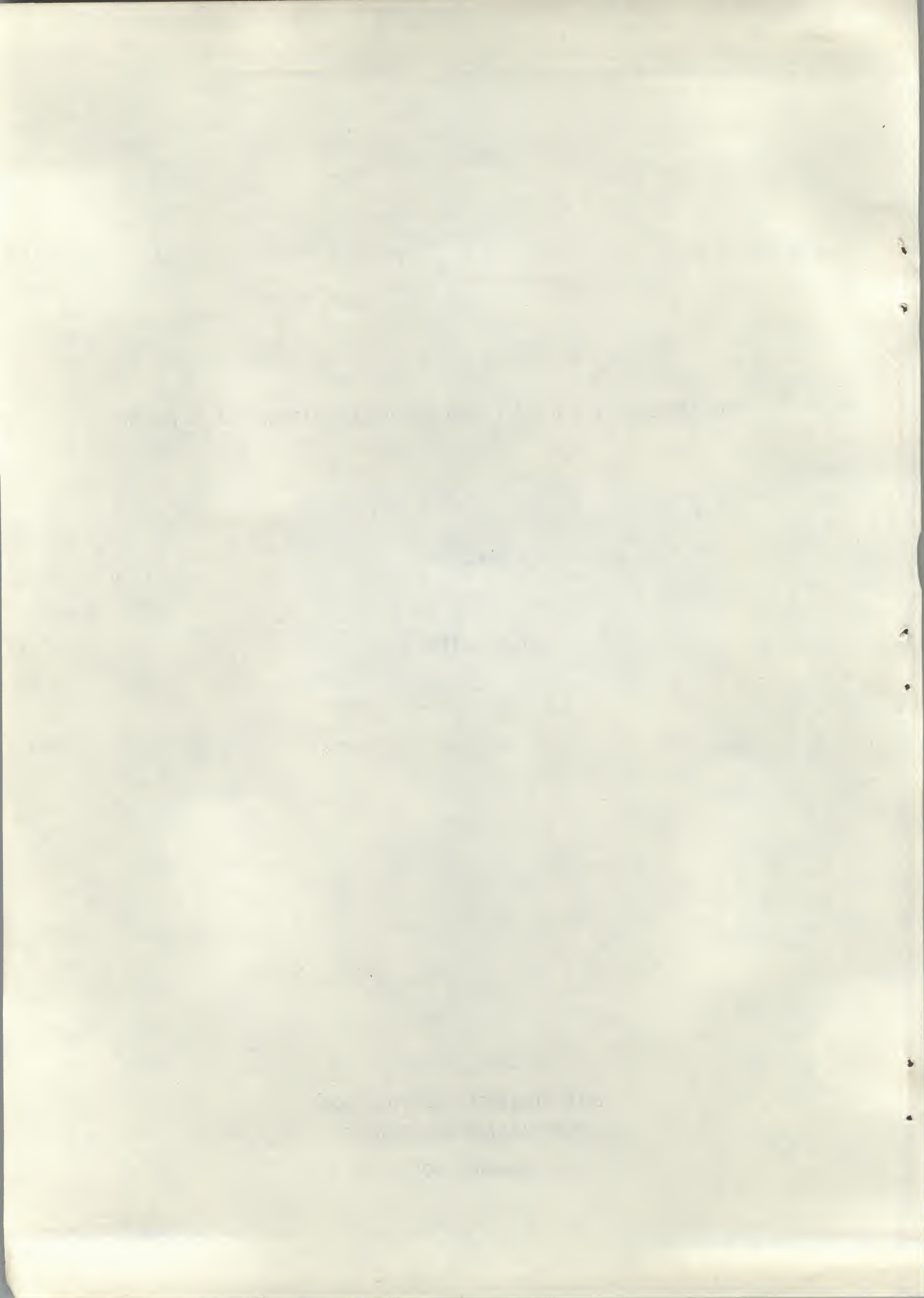
THE LANGUAGE M I D I A L AND ITS IMPLEMENTATION ON A 8K PDP-8.

- Part 1 -

H.A.M. Luijff

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF MATHEMATICS

February 1975



2. S U M M A R Y

The language MIDIAL and its implementation on a 8K PDP-8.

MIDIAL is one of the machine oriented higher level languages.

It contains a lot of features of ALGOL 68 and APL .

Some programming possibilities are :

- text editing and text preprocessing systems.
- description language for machine programs.
- language for teaching machine oriented programming.
- string and small integer arithmetical problems.

My dear Mr. [Name]
I have the honor to acknowledge the receipt of your letter of the 10th inst. in relation to the [subject] and in reply to inform you that the same has been forwarded to the proper authorities for their consideration. I am, Sir, very respectfully,
Yours, very truly,
[Signature]

3. THE CONTENTS.

1.	□	INTRODUCTION	1
2.	□	SUMMARY	2
3.	□	THE CONTENTS	3
4.	□	NOTATIONS	
4.1	□	FLOWCHART NOTATIONS	5
4.2	□	SYNTAX NOTATIONS	6
5.		MINIAL	
5.1	□	THE HISTORY OF MINIAL	10
5.2		LOADER SWAPPING AND MODIFICATIONS OF SYSTEM	
		SUBROUTINES	11
5.3		INTERFACING WITH THE OS/8 OPERATING SYSTEM	14
5.4		THE FILE COMMAND STRING FOR MINIAL	17
5.5		THE SYNTAX OF MINIAL	21
5.6	□	THE COMPILER OUTPUT CODE OR BINARY CODE	26
5.7		SYSTEM CREATION FROM BINARIES	30
6.		MIDIAL	
6.1	□	INTRODUCTION	31
6.2	□	THE SYNTAX OF MIDIAL	33
6.3	□	THE SEMANTICS OF MIDIAL	39
6.4	□	THE PREPROCESSOR	75
6.5	□	THE MIDIAL SYSTEM	91
6.6	□	SYSTEM CREATION FROM BINARIES	95
6.7	□	THE FILE COMMAND STRING	96
6.8	□	THE BINARY LOADER	102
6.9		THE BOOTSTRAP SYSTEM	105
	□	- THE LISTING OF THE PREPROCESSOR WRITTEN	
		IN MIDIAL	106

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

6.10	PROPOSALS FOR A NEW MIDIAL PREPROCESSOR	131
6.11□	EXAMPLES OF PROGRAMS WRITTEN IN MIDIAL	132
7.	□ REFERENCES	139
8.	APPENDICES	
8.1 □	LIST OF ERRORCODES	140
8.2 □	LIST OF IMPORTANT MACHINE ADDRESSES	144
8.3 □	THE SYSTEM SUBROUTINE ENTRY TABLE	146
8.4	CHANGING THE P1/P2 OUTPUT DEVICE	151
8.5 □	THE ASCII CODE TABLE	152
8.6	WORKING DOCUMENTS : PLANNING OF THE CREATION OF MIDIAL.	153.
8.7	WORKING DOCUMENTS : ANALYSIS OF THE PREPROCESSOR ACTIONS.	158

PART 2 : LISTINGS OF THE MINIAL SYSTEM .

PART 3 : LISTINGS OF THE MIDIAL PREPROCESSOR AND
COMPILER SYSTEMS .

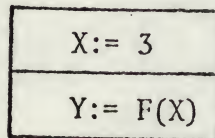


4.1 FLOWCHART NOTATIONS.

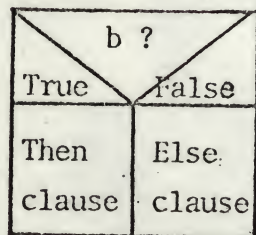
The flowchart technique for structured programming described by I.Nassi and B.Shneiderman (2) is used within this report.

There are four basic symbols be used. Combinations of them may be made to form structures, all of which are rectangular in shape.

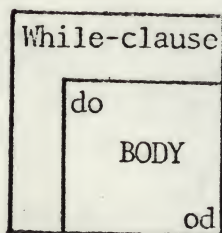
1. The process symbol : used to represent assignments, I/O operations and procedure calls. This symbol has the shape of a block.



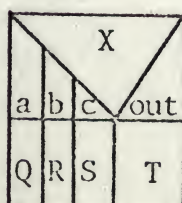
2. The decision symbol : is used for the representation of conditional clauses. (IF (b) THEN (a) ELSE (c) FI)



3. The iteration symbol : is used for the representation of looping statements as WHILE - DO clauses. The body of the iteration is a structure of arbitrary complexity.



4. The case symbol : is the symbol that represents (a) statement(s) by which the control of the program part is spread out over a number of possibilities depending on one control value, e.g. CASE x IN q,r,s : OUT t ESAC or IF x=a THEN q ELIF x=b THEN r ELIF x=c THEN s ELSE t FI .



THEORY OF THE EARTH

CHAPTER I. OF THE ORIGIN AND GROWTH OF THE EARTH.

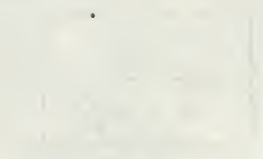
§ 1. The Earth is a sphere, and its surface is divided into two parts, the land and the water.

§ 2. The land is divided into continents, islands, and seas.

§ 3. The water is divided into oceans, seas, bays, and rivers.

§ 4. The land is divided into mountains, hills, and valleys.

§ 5. The water is divided into lakes, ponds, and streams.



§ 6. The land is divided into plains, mountains, and hills.

§ 7. The water is divided into rivers, lakes, and ponds.



§ 8. The land is divided into mountains, hills, and valleys.

§ 9. The water is divided into oceans, seas, bays, and rivers.

§ 10. The land is divided into mountains, hills, and valleys.



§ 11. The land is divided into mountains, hills, and valleys.

§ 12. The water is divided into oceans, seas, bays, and rivers.

§ 13. The land is divided into mountains, hills, and valleys.

§ 14. The water is divided into oceans, seas, bays, and rivers.

§ 15. The land is divided into mountains, hills, and valleys.



4.2 THE SYNTAX NOTATIONS.

The syntaxes within this report are set up in the ALGOL 68 notation way. (3)

The strict language is defined by means of a syntax and semantics.

This syntax is a set of production rules for notions.

It is expressed in small syntactical marks, in this report :

"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r"

"s","t","u","v","w","x","y","z" ; large syntactical marks :

"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R"

"S","T","U","V","W","X","Y","Z" ; and special marks :

":" (colon) , "," (comma) , ";" (semicolon) and "." (point) .

The representation of a terminal Ω symbol is the mark Ω :

At many times in the syntax the notation Ω is used directly instead of the much longer " Ω symbol" .

For example: the notion "a symbol" is directly replaced by the representation a .

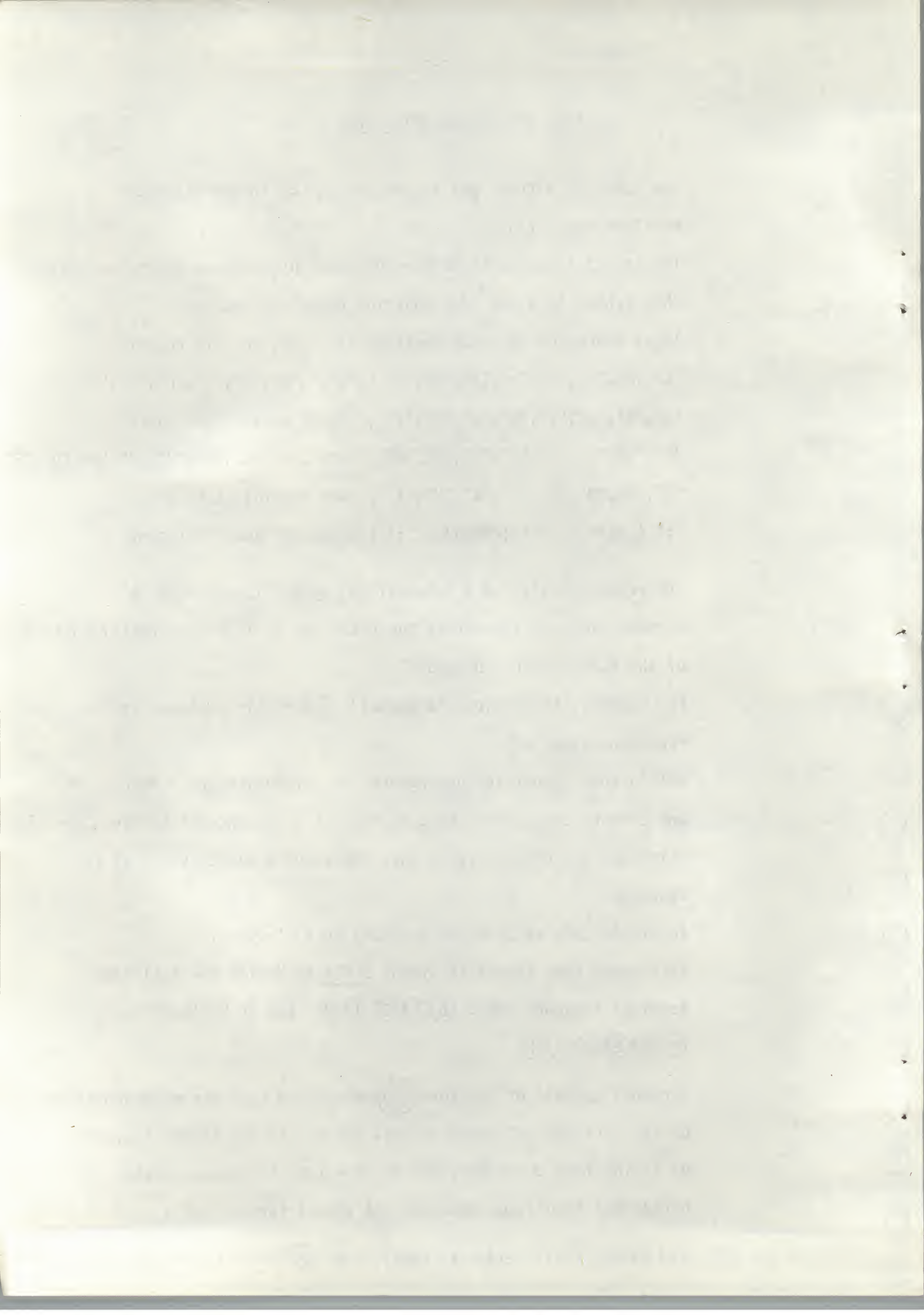
Within this report the underscored Ω symbol, where Ω represents one or more characters, must be read as a sequence of terminal symbols which are not underscored ; they represent a basic symbol of the language .

In MIDIAL this sequence is preceded by a ' symbol.

This means that the basic symbol BEGIN in MINIAL the following terminal sequence has : $\textcircled{B}\textcircled{E}\textcircled{G}\textcircled{I}\textcircled{N}$ and in MIDIAL this is :

$\textcircled{'}\textcircled{B}\textcircled{E}\textcircled{G}\textcircled{I}\textcircled{N}$.

Terminal symbols of the form $\textcircled{\Omega}$ mean for $\Omega = \text{cr}$ the representation of the carriage return character, for $\Omega = \text{lf}$ the terminal symbol of a line feed character, and for $\Omega = \text{tab}$ the representation of a horizontal tabulation character. A symbol terminal of a capital letter and of its small letter is equal, e.g. \textcircled{A} is equal to \textcircled{a} .



Within MIDIAL programs the terminal symbol sequence of basic symbols (underscored) may not be followed by a digit terminal symbol or a letter terminal symbol, with two exceptions :

- a. Is the length of a basic symbol smaller than two characters, then the terminal symbol sequence of that basic symbol may be followed immediately by a number consisting of maximal four digits.
So B 7766 can be written as the basic symbol B7766 .
- b. Are the lengths of the basic symbols in a sequence of basic symbols each smaller than two characters, they can be combined to a new basic symbol with as maximum the total length of six characters.
This means that the basic symbol sequence C AX X W 1 can be written as the basic symbol CAXXW1 with the terminal symbol sequence (C)(A)(X)(X)(W)(1) .

Note : The basic symbol V may be used only within a basic symbol combination if it occurs to the left hand side of other basic symbols, e.g. XVXX2 is allowed, but CXV not.

In MINIAL programs all basic symbols can be combined without restrictions.

A notion is a nonempty sequence of small syntactical marks, for which there is a production rule.

A production rule for a given notion consists of that notion followed by a colon, a list of notions and a point.

A list of notions is a nonempty sequence of members separated by commas.

A member is either a notion (productive) or is a terminal symbol or is empty .

The following is a list of the names of the persons who have been elected to the office of the President of the United States, from the year 1789 to the present time. The names are arranged in alphabetical order, and the year of election is given in parentheses.

George Washington (1789)
John Adams (1796)
Thomas Jefferson (1800)
James Madison (1808)
James Monroe (1816)
John Quincy Adams (1824)
Andrew Jackson (1828)
Martin Van Buren (1836)
William Henry Harrison (1840)
Francis Pickens (1856)
Abraham Lincoln (1860)
Andrew Johnson (1865)
Ulysses S. Grant (1868)
Rutherford B. Hayes (1876)
James A. Garfield (1880)
Chester A. Arthur (1881)
Grover Cleveland (1885)
Benjamin Harrison (1888)
William McKinley (1896)
Theodore Roosevelt (1900)
William Howard Taft (1908)
Woodrow Wilson (1912)
Calvin Coolidge (1924)
Herbert Hoover (1928)
Franklin D. Roosevelt (1932)
Dwight D. Eisenhower (1952)
John F. Kennedy (1960)
Lyndon B. Johnson (1964)
Richard M. Nixon (1968)
Jimmy Carter (1976)
Ronald Reagan (1980)
George H. W. Bush (1988)
Bill Clinton (1992)
George W. Bush (2000)
Barack Obama (2008)
Donald Trump (2016)

Some production rules are given explicitly and others are obtained with the aid of a metalanguage whose syntax is a set of production rules for metanotions.

A metanotion is a nonempty sequence of large syntactical marks.

A production rule for a given metanotion consists of that metanotion followed by a colon, a list of metanotions , and a point.

A list of metanotions is either a metanotion or a, possible empty, sequence of small syntactical marks.

The production rules for metanotions can be obtained from the rules given in the syntax for metanotions in the following step(s) :

If some rule contains one or more semicolons, then it is replaced by two new rules, the first of which consists out of the part of that rule up to and including the first semicolon with that semicolon replaced by a point, the second of which consists of a copy of that part of the rule up to and including the colon, followed by the part of the original rule following its first semicolon , whereupon this step is taken again .

The production rules for the strict syntax is any rule obtained in the following steps from the syntax rules defined in MINIAL or MIDIAL :

step 1 : The rule given above for metanotions is applied also for the strict syntax rules.

step 2 : One of the rules obtained is considered.

step 3 : If the considered rule contains one or more metanotions, then for some terminal production of such a metanotion , a new rule is obtained by replacing that metanotion , throughout a copy of the considered rule , by that terminal production , whereupon this new rule is considered instead .

and step 3 is taken ; otherwise, all blanks in the considered rule are removed and the rule so obtained is a production rule of the strict language .

The reader might find it helpful to read a ":" as "may be a" , a "," as "followed by a" , and a ";" as "or a" .

The semantics of the strict language :

A terminal production of a notion is considered as a linearly ordered sequence of symbols. This order is termed the "textual order", and following (preceding) stands for "textually immediately following" ("textually immediately preceding") in the syntax.

Typographical display features, such as blank spaces , change to a new line , and change to a new page do not influence this order.

Page 12 of 12

...

...

...

...

...

...

...

...

...

...

5.1 HISTORY OF THE LANGUAGE MINIALGOL (MINIAL)

Version 1 of the language was developed a couple of years ago, and provided most of the facilities available in the current version.

Version 2, which was developed also in Liverpool at the Department of Computational and Statistical Science, differs in some details of version 1. (1)

This version needs 4 K core memory fully . The monitor page is overwritten, while loading the MINIAL system and compiler in core. It was not possible to interface with I/O facilities other than they keyboard/printer and the highspeed papertape reader/punch . To compile and to load and execute, the programmer had to be in the neighbourhood of the computer. There was no possibility to compile and run programs at a terminal in another room.

This part of this report treats the changes made to the Liverpool-2 MINIAL version, which is base of our system.

First of all the swapping of the loader and the monitor is treated, together with the simplifications made to the compiler and the system subroutine package.

Secondly the interfacing with the OS/8 operating system is treated, in particular the input and output via device handlers.

THE HISTORY OF THE

... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...
... of the ...
... of the ...
... of the ...
... of the ...
... of the ...
... of the ...

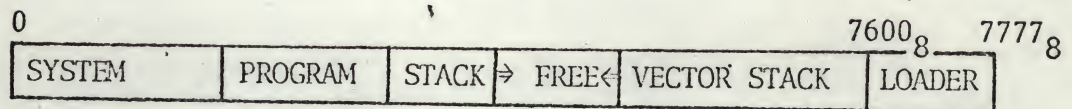
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...

... of the ...
... of the ...

5.2 LOADER SWAPPING and MODIFICATIONS OF THE SYSTEM SUBROUTINES.

In the Liverpool-2 version the store lay-out is as following :



About 1100 locations contain the system-subroutines, which provide the arithmetic and data manipulative facilities. (1)

Above these locations (a) program(s) , in particular the compiler, can be loaded with the system loader, which resides in the last page of the memory field.

Before the first changes, this Minial implementation version had to be loaded with two papertapes every time. After the running of Minial, a new usage of the computer could be made only after bootstrapping the systems monitor in core.

The first change was extending the Minial system with a once only program and a start program part.

With the once only program, the monitor page of the memory field, in which the Minialgol system had to be loaded, was saved in another memory field. Then the Minial system was loaded and the loader was swapped afterwards with the monitor page.

The system could be saved then, after returning to the monitor.

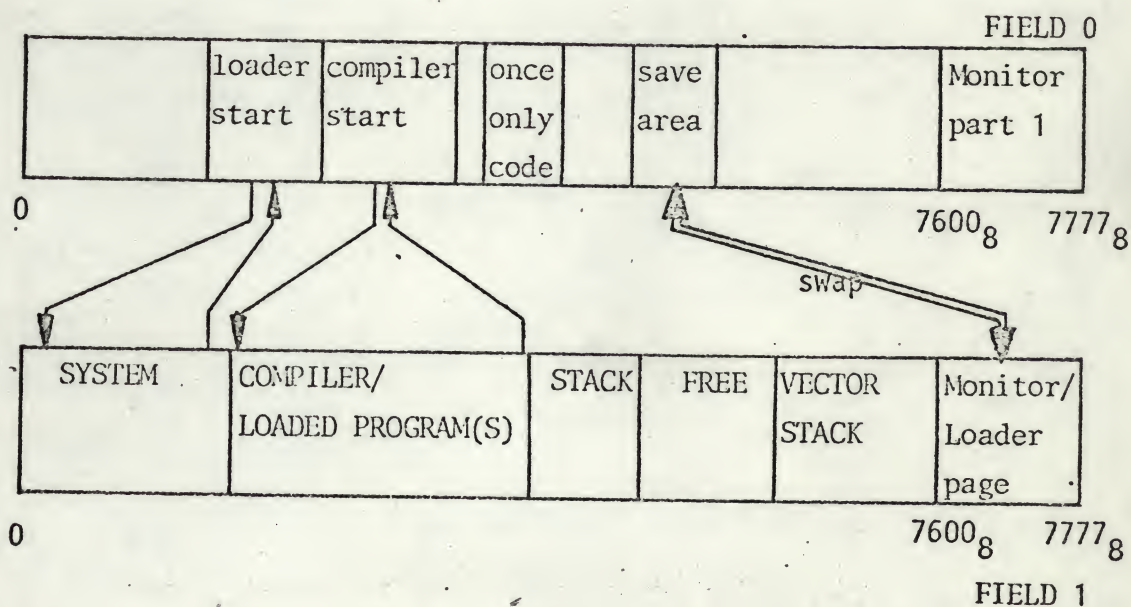
Starting up on the start program part, the loader was swapped in its old locations (the monitor page), and the system was started at its old start address.

The Liverpool-2 compiler compiled a program and halted afterwards, waiting on a new program to compile. From there it was possible to start up the system loader by toggling in the loader start address and restarting from there.

THE HISTORY OF THE
CITY OF BOSTON

From the first settlement of the city in 1630 to the present time. The history of the city of Boston is a story of growth and development. It begins with the arrival of the Puritans in 1630, who sought a place where they could practice their religion freely. They found a rugged and cold land, but they were determined to build a city. Over the years, the city grew from a small settlement to a major center of commerce and industry. It was a city of innovation and progress, a city that led the way in many fields. The history of the city is a testament to the spirit of the American people, a spirit of freedom and independence. It is a story that is as relevant today as it was in the past. The city of Boston is a city of many firsts, a city that has shaped the course of American history. It is a city that is proud of its past and looks forward to its future. The history of the city is a story that is still being written, a story that is as exciting as ever.

In the first Delft version this was changed into a compiler which returned to the start program, swapped back the loader and returned to the monitor. Afterwards the system could be started from the monitor at a second start point, which entered the loader and the run time system. We got now the following store lay-out: (not scaled)



When we got the source listings of the subroutine system and of the compiler from Liverpool, we built new sources while throwing away the overflow tests in the arithmetical routines.

The ' operator (addition without overflow testing) is replaced by the + operator. The double length operators (MULT, DIV and ADD) were shortened to their single length equivalents (*, / and +).

The changes made to the compiler were :

1. The input device was changed from channel 1 (keyboard) to the high-speed papertape reader (channel 3).
2. At compiling a program ; a listing is made of the inputted program on channel 1. (printer of the teletype)

The error printing routine could be shortened because of the fact that a line in error did not to be printed.

Handwritten text at the top of the page, likely a title or introductory paragraph, which is mostly illegible due to fading.



Handwritten text in the middle section of the page, continuing the narrative or providing details related to the diagram above.

Handwritten text at the bottom of the page, which appears to be a concluding paragraph or a signature block.

These modifications resulted in a saving of about 100 core locations.

After joining the Input/Output package and the system driver, the following changes were made:

1. The Halt routine or error routine was changed so that the system control returns from this routine to the monitor, after closing the output file and printing the text : "SYS ERR AT".
2. Within the compiler the papertape leader output (64 null characters) was removed.
3. The following returns to the I/O package were inserted :
 - a. at the compiler end, the control is returned to the I/O pack/ system driver.
 - b. within the loader, a subroutine call is inserted to the I/O pack/ system driver to change the control mode from load to execute.
 - c. At the program end, a jump is made to the system driver.
4. The channel 3/4 device was changed from the high-speed papertape punch to the buffered output device given in the command string or to its default device (SYS), and for input the channel 3/4 device was changed from the high-speed papertape reader to a buffered input stream from the given input device in the command string or its default value.

5.3 THE INTERFACING WITH THE OS/8 OPERATING SYSTEM.

To get a more usable compiling and run time system for MINIAL, it was necessary that for the highspeed input and output operations all, to the computer coupled, I/O devices could be used.

This is done by creating a program part which interferes with the OS/8 operating system for input and output. (6)

In this program part, called here further I/O pack, the input-/output file controls are done and also the general compiler and load/execute control is done.

The OS/8 operating system has a part, called the User Service Routines or shortly the USR (PS/8 Handbook, chapter 5), by which it is possible to search a file on a given device, to open an output file, to chain to another saved program, etcetera.

The file command string (page 17), typed in after the asterisk of the Command Decoder (PS/8 Handbook, chapter 3) which treats this intyped string, is used for defining all the input and output files; or devices, which the system must handle.

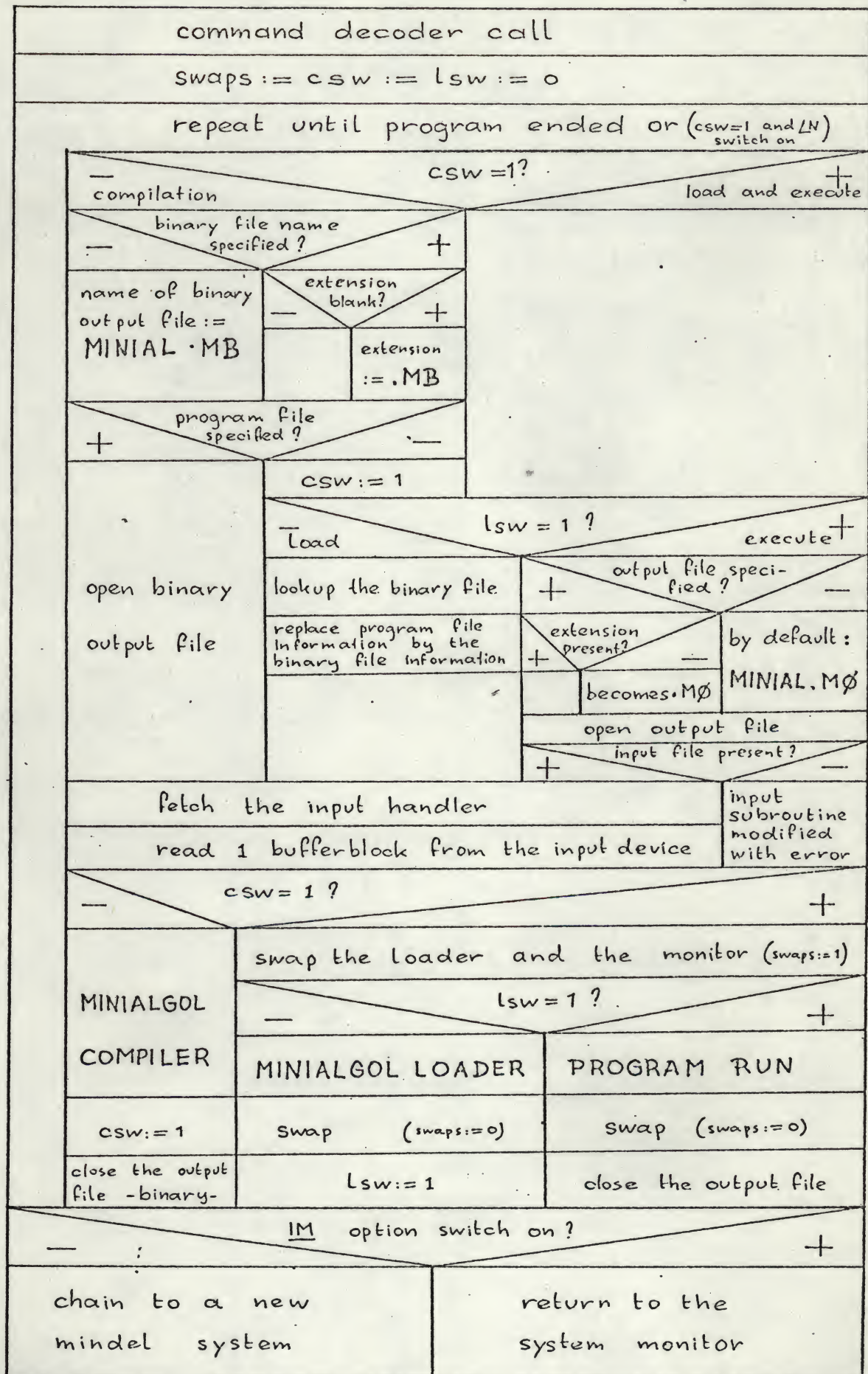
Also option switches can be specified, these are used for changing the normal system control.

The /N switch modifies the system so that only the compilation phase is executed, and that the control returns to the Command Decoder or the systems Monitor, depending on the /M option switch.

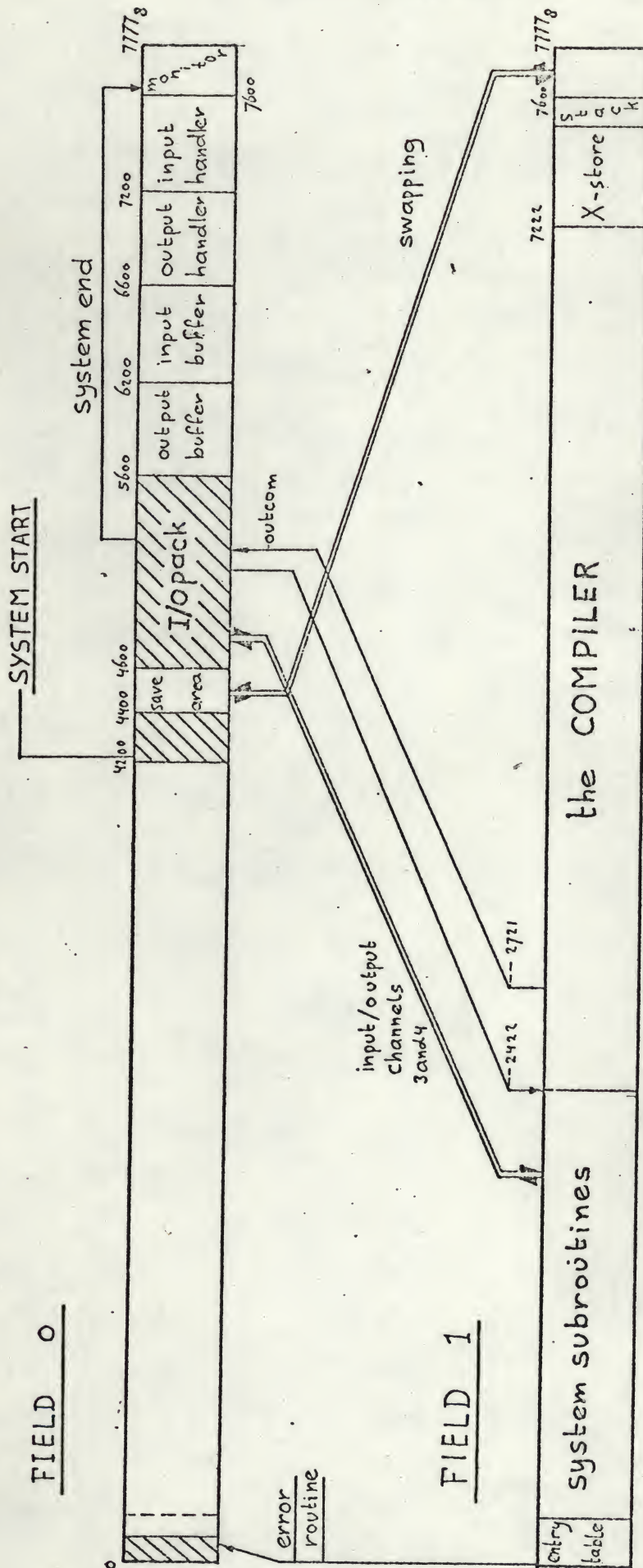
On the next page the flowchart of the MINIAL I/O- and control system is given.

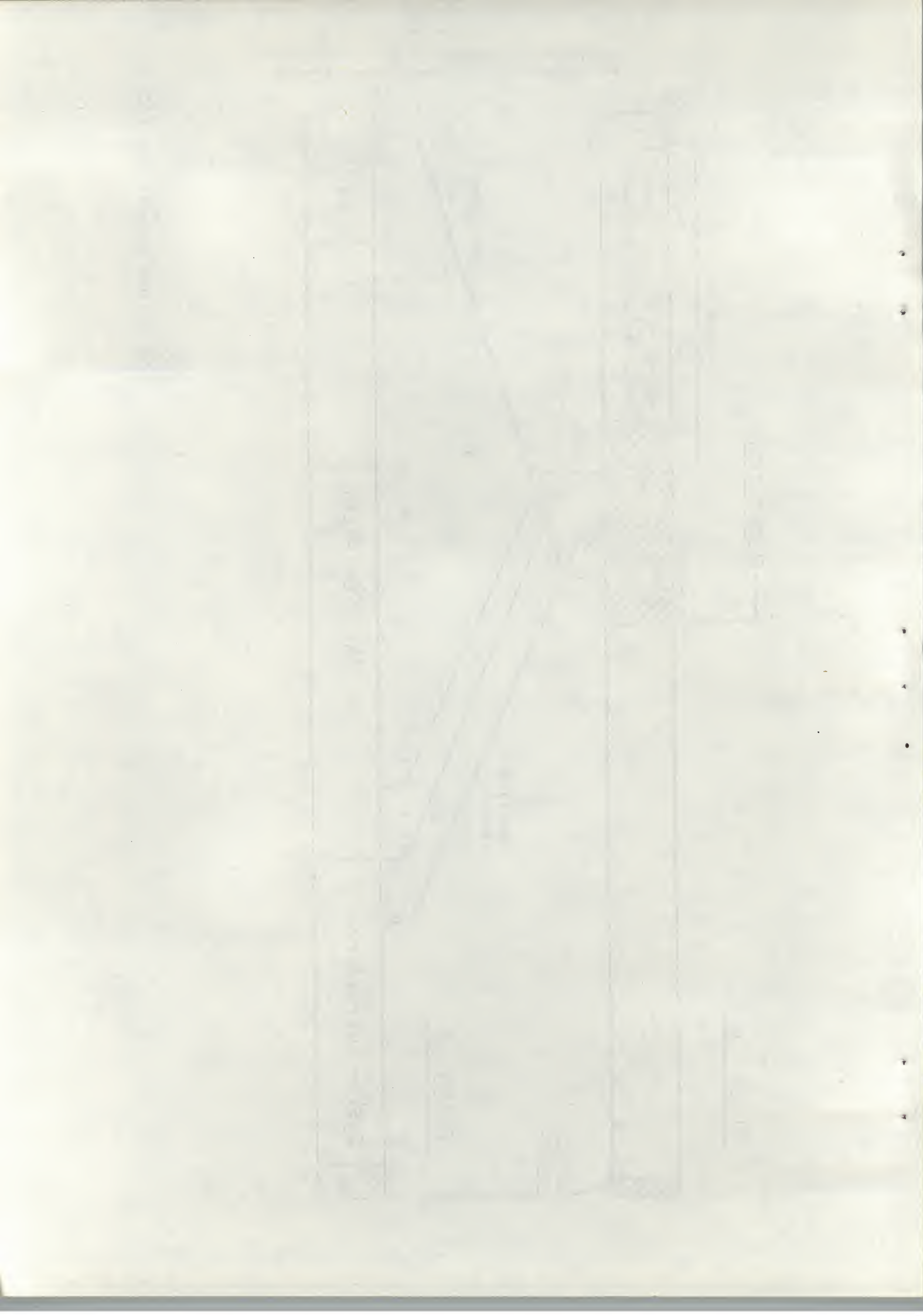
The I/O pack is located in the memory locations 4200 till 5577₈ of the memory field 0. From 5600 till 6577₈ the input and output buffers are placed, and from 6600 till 7577₈ the input and output handlers (2 pages each) are located.

By changing some assembly parameters, it is possible to relocate the system parts through the whole field 0.



MINIAL SYSTEM





5.4 THE FILE COMMAND STRING FOR THE MINIALGOL SYSTEM.

The Minialgol system asks the user to identify which files must be used by the compiler and the loader.

The user must also identify whether the Minialgol system must only compile, or must do the load- and go cycle only, or must do a compile- load- and go cycle.

The command string must be typed in by the user after the * of the Command Decoder :

```
*{<dev>:}{BINFILE{.ex1}} {,{<dev>:}{OUTFILE{.ex2}} < }
    {<dev>:}{PROGRAM{.ex3}} {,{<dev>:}{INFILE{.ex3}}}{/M}{/N} +
```

{...}

- means that this sequence can be omitted, and that this sequence will be replaced by the Minialgol system by a default sequence in some cases.

{<dev>:}

- is the codeword for specifying an input or output device.
- if it is not specified, the DSK device is the default value for the device code.¹⁾

{BINFILE{.ex1}}

- BINFILE.EX is the file in which the compilation result is set. ('binary code')
This result file is also the file that becomes loaded by the loader.
- If the BINFILE is not specified by the user, the default name of the file is :.

THE HISTORY OF THE

REIGN OF

CHARLES THE FIRST

BY

JOHN BURNET

OF

THE UNIVERSITY OF OXFORD

IN TWO VOLUMES

LONDON

Printed by J. Streater, at the Sign of the Gun, in St. Dunstons Church-yard

1679

THE HISTORY OF THE

REIGN OF

CHARLES THE FIRST

BY

JOHN BURNET

OF

THE UNIVERSITY OF OXFORD

IN TWO VOLUMES

LONDON

Printed by J. Streater, at the Sign of the Gun, in St. Dunstons Church-yard

1679

THE HISTORY OF THE

REIGN OF

CHARLES THE FIRST

BY

MINIAL.MB .

{.ex1}

- If the extension (ex1) is not specified, it will have the default value of .MB .

{OUTFILE{.ex2}}

- OUTFILE.EX is the file name of the file on which the output at run time is written by by the P3 and P4 instructions of Minialgol.
- If the file name is not specified, then the file MINIAL.MO is generated by default, if there has been output on P3 or P4.

{.ex2}

- If the extension (ex2) is not specified, it gets the default value of .MO .

{PROGRAM{.ex3}}

- PROGRAM.EX is the name of the file that must be compiled. 2)

load-and go

- If the file name and the device are not specified, then the file on the BINFILE place (or its default value MINIAL.MB) must be loaded and run.

{INFILE{.ex3}}

- INFILE.EX is the file name of the file where the data , at the R3 or R4 instruction is read from. 2)
- If the device code and the file name are not specified, the system returns after an R3 or R4 instruction to the Monitor , with the error printout 'I/O ERR AT nnnn'.



{.ex3}

- The default value for the extension of the input files is .MI (Minialgol Interpreter):

If the user does not specify an extension for an input file, the default value .MI is added to the file name. If the file is not found, a search for a file without an extension to the file name is made.

{/M}

- is specified in case that the control of the system must return, after completion, to the Monitor.

In the case that the file MINIAL.SV is not on the SYS(-tem) device, this option switch MUST be specified. (else you get a monitor error)

- If this switch is not specified, the general control returns to a fresh Minialgol system. This is done by chaining to the MINIAL.SV file on the system device.

The system returns with the * of the Command Decoder.

{/N}

- This is the No load switch, to do only (a) compilation(s).

This option can be used independently of the /M option.

only compilation

- 1) If the input device of the INFILE.EX is not specified, the input device of the PROGRAM.EX file is taken as default input device of the INFILE- file.



In case of only the load- and go cycle, then the device code of the INFILE- file must be specified.

- 2) In case that an input device is a non-directory device , e.g. the high-speed papertape reader and the teletype, the file name can be omitted.
- 3) Instead of the < sign, the ← sign can also be used.



5.5 METANOTIONS FOR THE MINIALGOL SYNTAX.

ALPHA :: a;b;c;d;e;f;g;h;i;j;k;l;m;n;o;p;q;r;s;t;u;v;w;x;y;z
NOTION :: ALPHA ; NOTION ALPHA .

META PRODUCTION RULES.

EMPTY *non-ε* :
NOTION option : NOTION ; EMPTY .
NOTION sequence : NOTION ^{concat} ; NOTION ^{div} ; NOTION sequence .
NOTION list : NOTION ; NOTION , (,) , NOTION list .
NOID : nonvoid ; void .
Ω symbol : (Ω) .
comma NOTION : comma , NOTION .
NOTION comma : NOTION , comma .

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO LIBRARY

540 EAST 57TH STREET, CHICAGO, ILL. 60637

ACQUISITIONS

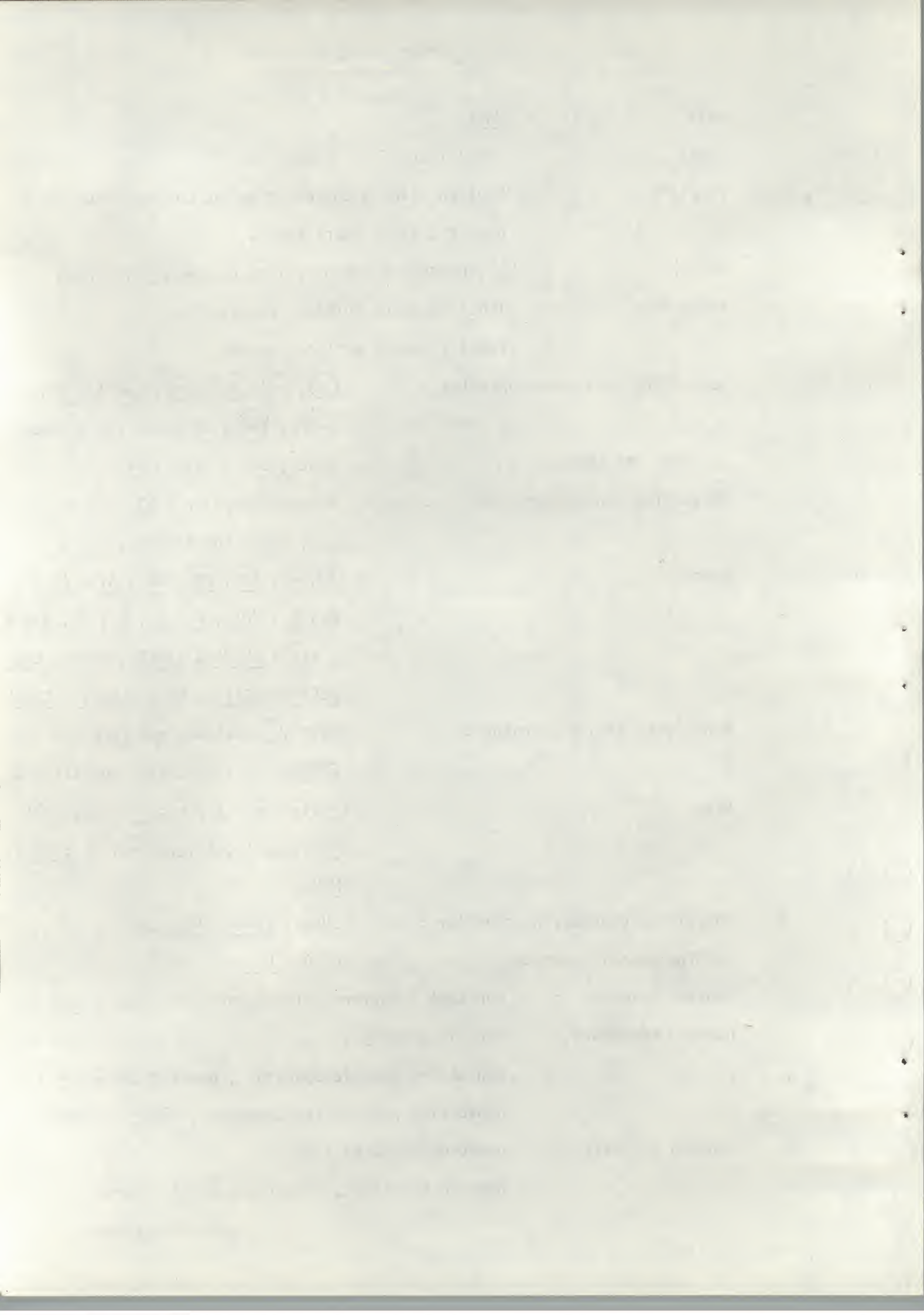
For information on the University of Chicago Library's
policies and procedures, please contact the
Acquisitions Department at (773) 936-7200.
The University of Chicago Library is a
member of the Association of American
University Libraries (AAUL) and the
Association of Research Libraries (ARL).
For more information, please visit our
website at <http://www.lib.uchicago.edu>.

PRODUCTION RULES .

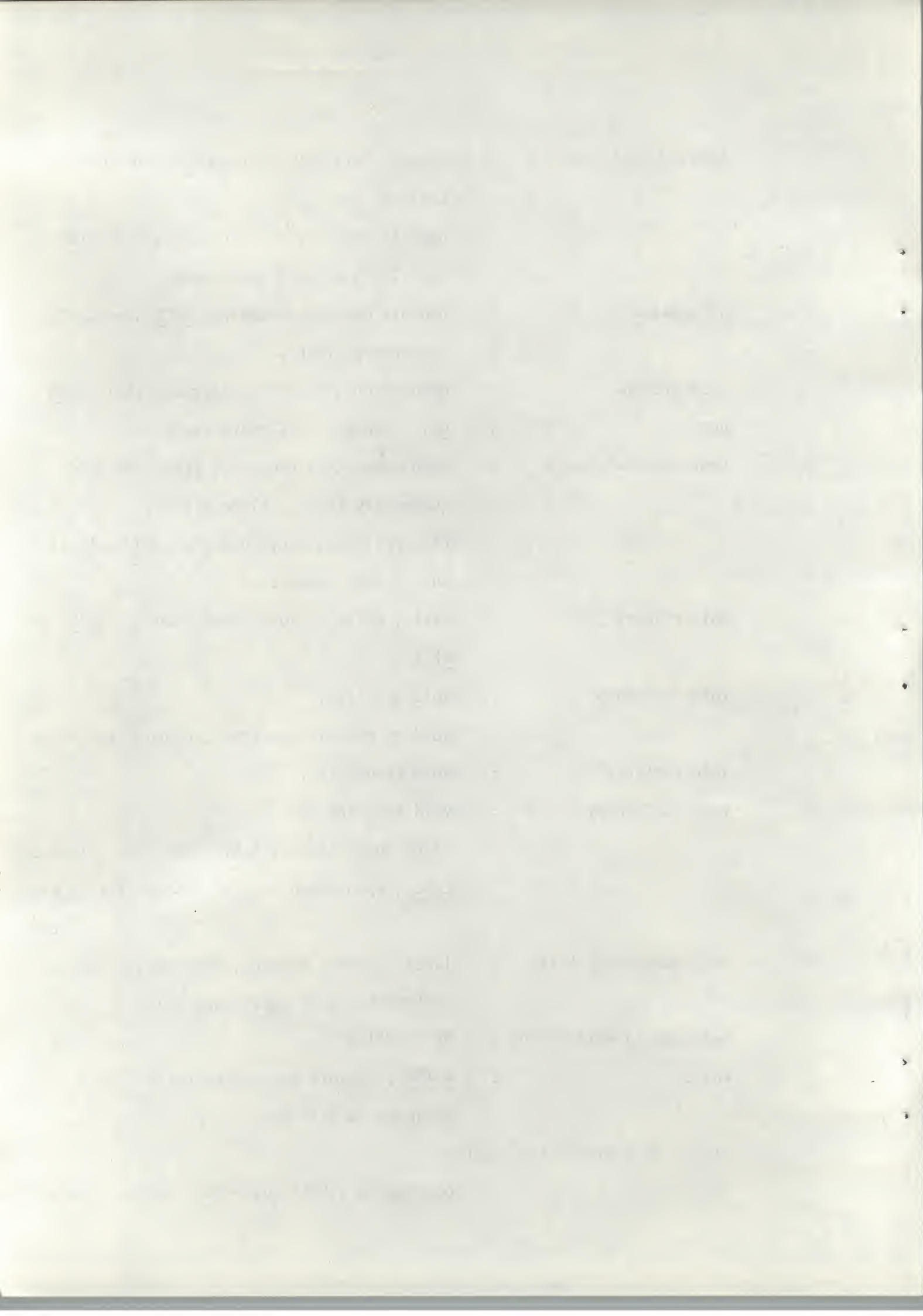
letter	: ALPHA symbol .
octal digit	: ①; ②; ③; ④; ⑤; ⑥; ⑦ .
digit	: octal digit ; ⑧; ⑨ .
comma	: , .
quote	: " .
quote image	: quote , quote .
hek	: # .
escape	: % .
space	: .
skip	: cr symbol ; lf symbol ; tab symbol .
character	: letter ; digit ; comma ; & ; ! ; (;) ; ' ; : ; ; ; / ; ? ; + ; = ; . ; - ; * ; [;] ; \$; \ ; @ ; ↑ ; ← ; < ; > .
mark	: character ; quote ; hek ; escape .
string token	: character ; hek ; quote image ; escape ; space .
comment token	: character ; quote ; escape ; space ; skip .
general character	: mark ; skip ; space .
octal integer	: octal digit sequence .
number	: digit sequence .
label identifier	: <u>L</u> , number ; <u>G</u> , number .
octal value	: <u>B</u> , number .
boolean value	: <u>TRUE</u> ; <u>FALSE</u> :
character value	: escape , mark ; <u>SP</u> ; <u>LN</u> .
string	: quote , string token sequence option , quote .
comment	: hek , comment token sequence option , hek .
open token	: (; <u>IF</u> ; <u>CASE</u> ; <u>BEGIN</u> .
close token	:) ; <u>FI</u> ; <u>ESAC</u> ; <u>END</u> ; <u>OD</u> .



exit : HALT .
 label : label identifier , \odot .
 constant : boolean value ; character value ; octal value ;
 number ; label identifier .
 vector : \odot , nonvoid quaternary list option , \odot ; string .
 table item : label sequence option , constant ;
 label sequence option , string .
 nonvoiding assignment operator : $\odot \leftarrow$; $\odot =$; $\odot * \leftarrow$; $\odot * =$; $\odot \leftarrow @$; $\odot = @$;
 $\odot * \leftarrow @$; $\odot * = @$; operator and becomes .
 operator and becomes : dyac , $\odot \leftarrow$; dyac , $\odot =$.
 nonvoiding monadic operator : monop , $\odot @$ option ; ST , $\odot @$;
 $\odot \dots$, label identifier , $\odot \dots$.
 monop : $\odot +$; $\odot -$; AP ; AR ; AW ; AY ; AZ ; C
M ; N ; NOT ; P ; R ; V ; W ; X ; Y
Z ; $\odot \$$; ALPHA ; DIGIT ; CHAR ; DECS
DECIN ; OCTIN ; OCTS ; COPY ; CALLP
 nonvoiding dyadic operator : dyac , $\odot @$ option ; $\odot +$; COMP , $\odot @$;
EXPAND , $\odot @$; $\odot \dots$, label identifier , $\odot \dots$.
 dyac : $\odot +$; $\odot -$; $\odot *$; $\odot /$; $\odot >$; $\odot \geq$; $\odot =$; $\odot / =$;
 $\odot \leq$; $\odot <$; $\odot !$; $\odot \&$; $\odot \cdot$; $\odot \leftarrow$; SL ; HD ;
MEM ;
 nonvoiding proceduring operator : SUBR ; EXPR ; PROG .
 voiding monadic operator : GOTO ; DCL .
 nonvoid primary : constant ; vector ; closed nonvoid clause ; SREG .
 nonvoid secondary : nonvoid primary ;
 ; nonvoiding monadic operator , nonvoid secondary ;
 nonvoiding proceduring operator , NOID secondary .
 nonvoid tertiary : nonvoid secondary ;
 nonvoid tertiary , nonvoiding dyadic operator ,
 nonvoid secondary .



nonvoid quaternary : nonvoid tertiary ; closed nonvoid quaternary
 train ;
 nonvoid tertiary , nonvoiding assignment
 operator , nonvoid quaternary .
 if chooser : nonvoid quaternary train , THEN , nonvoid
 quaternary train .
 case prelude : open token , nonvoid quaternary train , IN .
 out : OUT , nonvoid quaternary train .
 nonvoid conditional : open token , if chooser , ELSE , nonvoid
 quaternary train , close token ;
 case prelude , nonvoid quaternary train list ,
 out , close token .
 void primary : exit ; while ; closed void clause ; RESET ;
SKIP .
 void secondary : void primary ;
 voiding monadic operator , nonvoid secondary .
 void tertiary : void secondary .
 void quaternary : void tertiary ;
TABLE , open token , table item list , close token
CODE , open token , octal integer list , close
 token
 NOID quaternary train : label sequence option , NOID quaternary ;
 statement , NOID quaternary train .
 balanced to void NOTION : NOID NOTION .
 while : WHILE , nonvoid quaternary train , DO
 balanced to void quaternary .
 closed NOID quaternary train :
 open token , NOID quaternary train , close token



NOID quaternary trist : nonvoid quaternary trist comma option ,
NOID quaternary train , comma balanced to void
quaternary trist option .

void conditional : open token , if chooservoid , else option ,
close token ;
case prelude , balanced to void quaternary trist
close token ;
case prelude , void quaternary trist , OUT ,
nonvoid quaternary trist , close token .

if chooservoid : nonvoid quaternary train , THEN , balanced to
void quaternary train .

else : ELSE , void quaternary train .

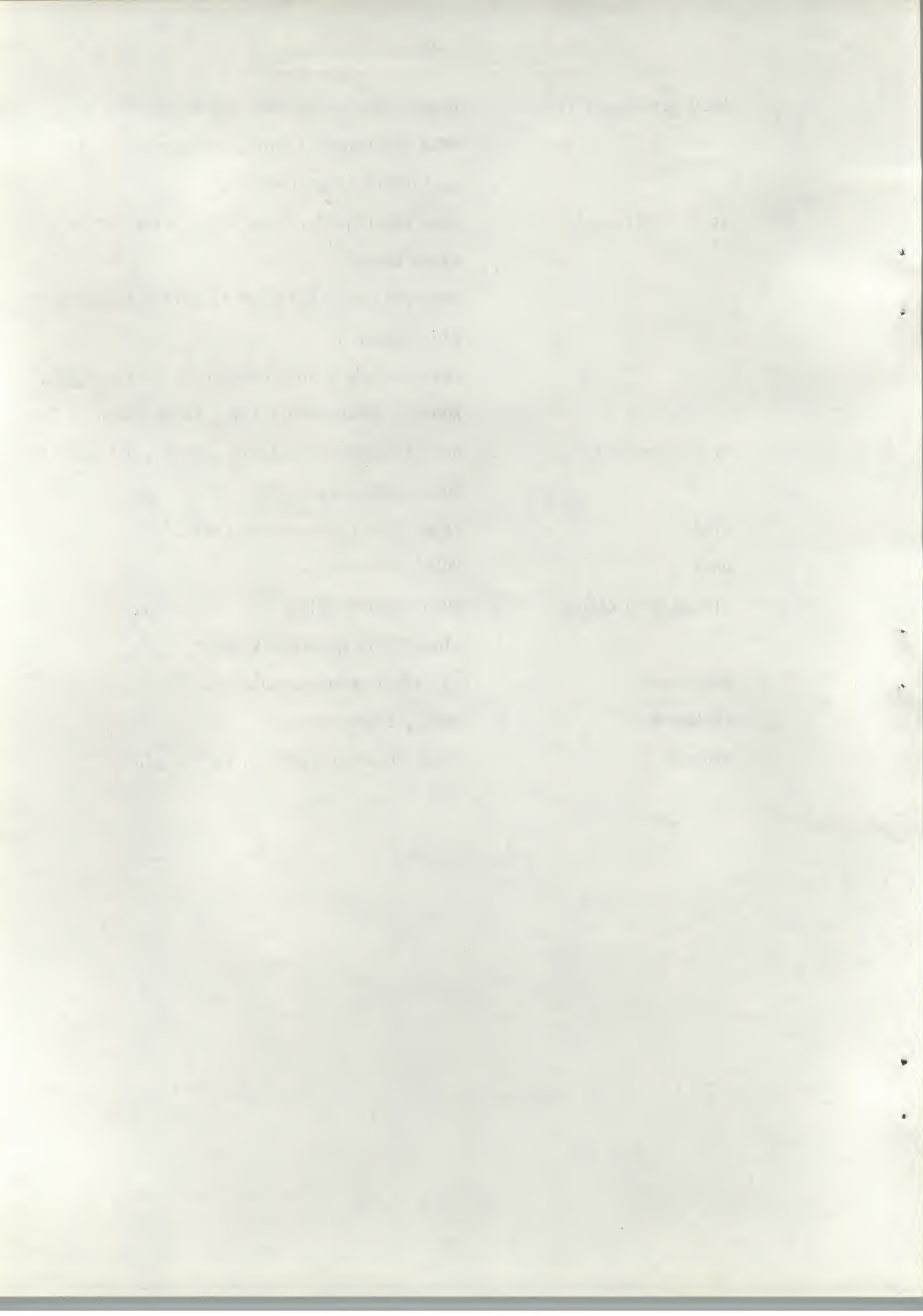
unit : NOID quaternary .

closed NOID clause : NOID conditional ;
closed NOID quaternary train .

sequencer : (;, label sequence option .

statement : unit , sequencer .

program : label sequence option , unit , (;).



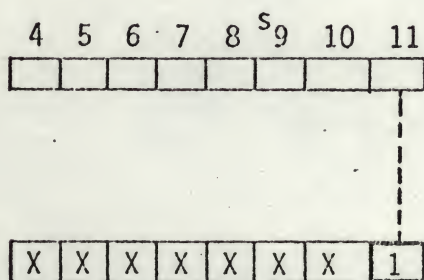
5.6 THE COMPILER OUTPUT CODE OR BINARY CODE.

The binary code consists out of 8 bit words or 'frames', this because of the fact that the first Minial version a system was with papertape as output.

This code is handy to handle it as ASCII- data on the input and output devices by means of the device handlers.

The first dataword of the code string is 125_8 .

Here below an explanation of the code is given for all the combinations which the loader can handle. This does not mean that the compiler can generate all these codes, for example: the recursive loader call can not be generated by the compiler.



- this is the bit number of the accu or frame on papertape.
s = sprocket hole of papertape.

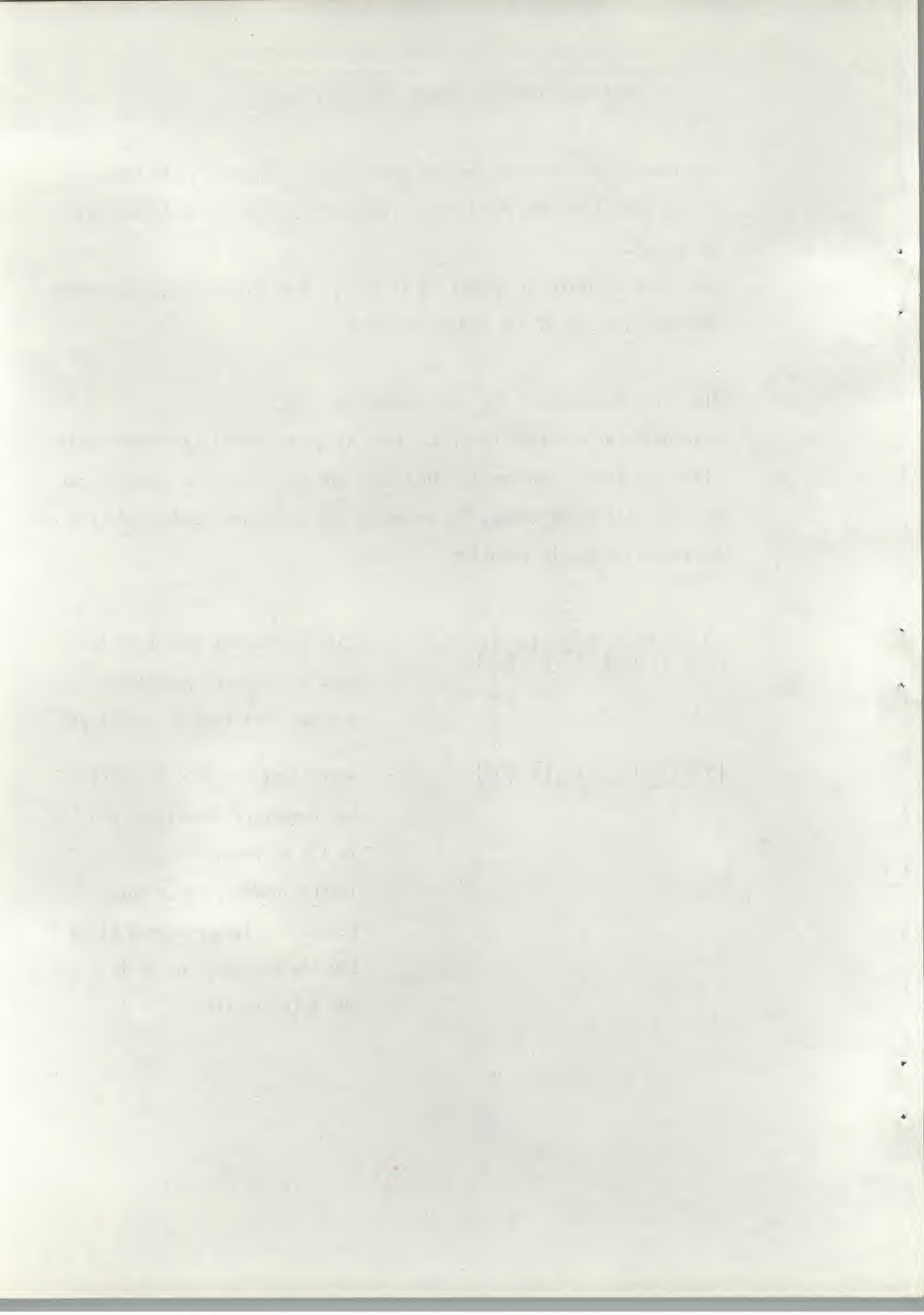
- means that bit 4 to 10 specify the number of the system routine in the subroutine table.

In the loaded program this

becomes : (after adding B4400)

100 10x xxx xxx₂ or as in PAL-8 :

JMS Z (x xxx xxx .



4 5 6 7 8 ^s9 10 11

L	L	L	L	L	L	0	0
---	---	---	---	---	---	---	---

H	H	H	H	H	H	0	0
---	---	---	---	---	---	---	---

0	0	0	0	.	.	1	0
---	---	---	---	---	---	---	---

0	0
---	---

0	1
---	---

1	0
---	---

1	1
---	---

- bit number

- is the first word of a combination of two frames . It is followed by:

and these two words are combined to a full twelve bit word as following:
HHH HHH LLL LLL₂ .

This is used for the output of constants, data after the CODE operator and for operation codes as RESET , SKIP , LN and SP.

- program directive.

- end of program, signals the loader end also.

- recursive loader call .

- absolute data follows in the next two frames as twelve bit data.

- relative data follows in the next two frames as twelve bit data.

These last two directive words and their following data words are followed by another two frames data word. This denotes the relative address from the load buffer begin where the data must be filled in. If it is absolute, the data is filled in without else with the addition of the buffer start address.

1900

1901

1902

1903

1904

1905

1906

If the value of the accessed location was not zero at that time, the data word is filled in at this location, and the value which was in that accessed location is used as a backreference, with as base the load buffer begin to locations where that data word must be filled in.

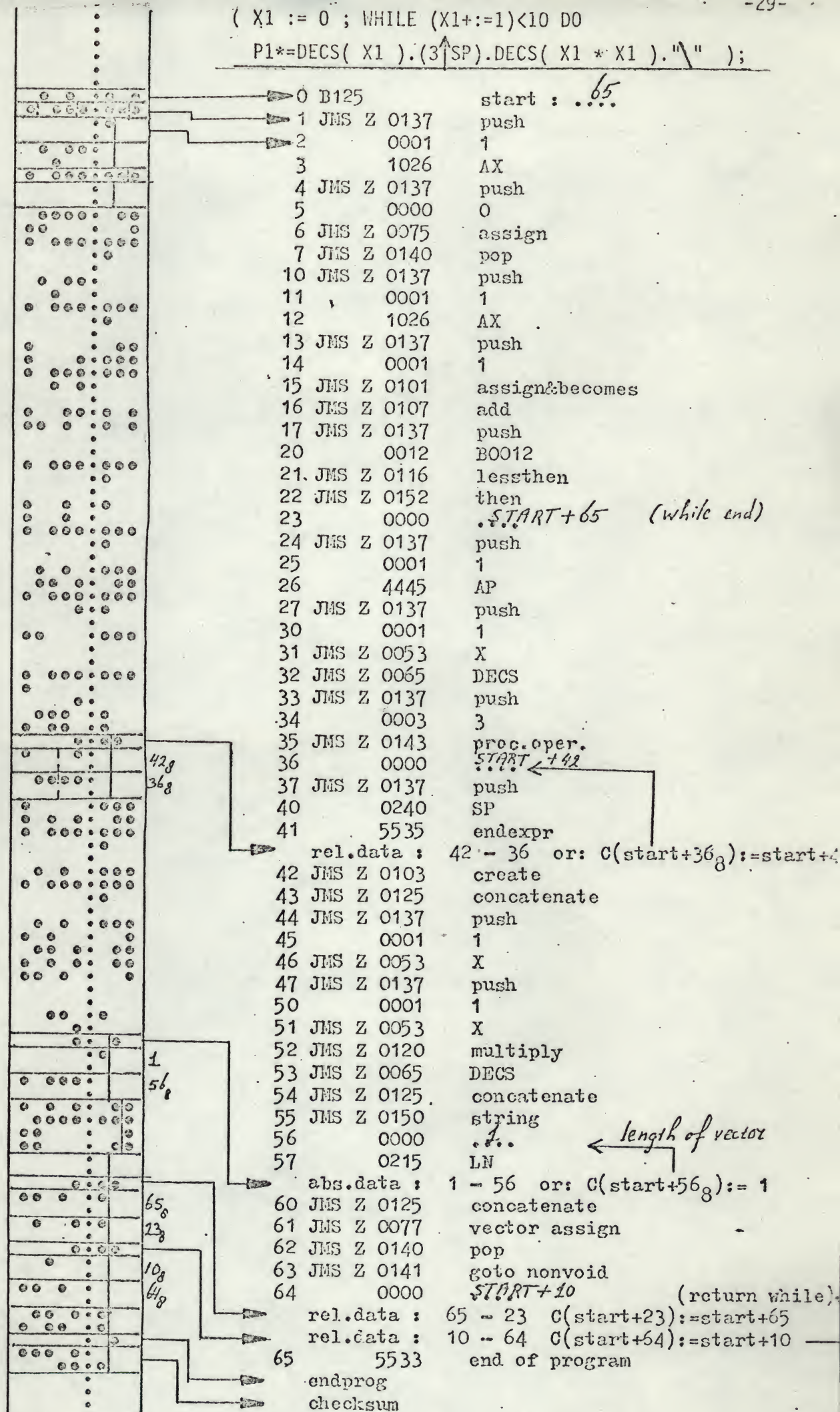
This is especially used in loading of a program with labels that are used before their actual declarational usage.

On the next page an compiled example is used to demonstrate the compiler output code, and the directive frames.



(X1 := 0 ; WHILE (X1+=1)<10 DO

P1*=DECS(X1).(3↑SP).DECS(X1 * X1).\");





5.7 THE MINIAL.SV CREATION FROM BINARIES.

Creating a new Minial system, the programmer needs the PAL8 binaries of the Minial system subroutines (MINISY.BN) and of the Minial compiler (MINICM.BN) .

To create the Minial save system the programmer must type in:

```
↑C
.RU dev·ABSLDR ↑
*dev : MINISY.BN ↑
*dev : MINICM.BN $
.SAVE dev MINIAL 0-177,4200-5577,10000-16777 ; 4200=0 ↑
```

↑ stands for the carriage return character,

\$ for the altmode character.

For dev the input or output device code must be filled in.



6.1 MIDIALGOL (MIDIAL) - INTRODUCTION.

Proceeding from the Delft MINIAL version, the language MINIAL is extended to the language MIDIAL (GOL) .

This was done because of the facts that:

1. MINIAL had no identifiers, only X, Y and W- stores.

In MIDIAL it is possible to give symbolic names to the parameters of subroutines and to variables.

2. The usage of quaternary trains was not used consequently within the DO-clause as in the THEN-clause.

The MINIAL construction WHILE .. DO (...;...;...) ; is written in MIDIAL as WHILE .. DO ...;...;... OD ; .

3. A fuction call in MIDIAL is written as a fuction in the ALGOL way:

$F(X,Y)$ is equivalent to the MINIAL function call $[G_f, X_x, X_y]$.

4. The MINIAL system was not interruptable, the MIDIAL system is made interruptable so that the control over "repeat forever" and other programs returns to the monitor after typing the $\uparrow C$ (control C) character on the keyboard of the system teletype.

Still some other changes are made as for example: VEC(...) which is an equivalent of the vector [...] in MINIAL .

To compile MIDIAL programs a preprocessor, written initially in MINIAL, is used together with a minimal subroutine package (system), which is based upon the old MINIAL system subroutine package.

Together they form the preprocessor system. (MIDIAL.SV)

After the testing period of this system, a minimum preprocessor was created to preprocess a preprocessor written in MIDIAL itselfs.

THE HISTORY OF THE

REIGN OF KING CHARLES THE FIRST

IN THE YEAR 1649

BY JOHN BURNET

IN TWO VOLUMES

LONDON

Printed by J. Sturges, at the Sign of the Crown, in St. Pauls Church-yard

1724

IN TWO VOLUMES

LONDON

Printed by J. Sturges, at the Sign of the Crown, in St. Pauls Church-yard

1724

IN TWO VOLUMES

LONDON

Printed by J. Sturges, at the Sign of the Crown, in St. Pauls Church-yard

1724

IN TWO VOLUMES

LONDON

Printed by J. Sturges, at the Sign of the Crown, in St. Pauls Church-yard

1724

IN TWO VOLUMES

LONDON

Printed by J. Sturges, at the Sign of the Crown, in St. Pauls Church-yard

1724

This minimum preprocessor is called the bootstrap preprocessor.

After processing a program, the preprocessor system chains to a compiler and run-time system, which is based upon the old Minial system.

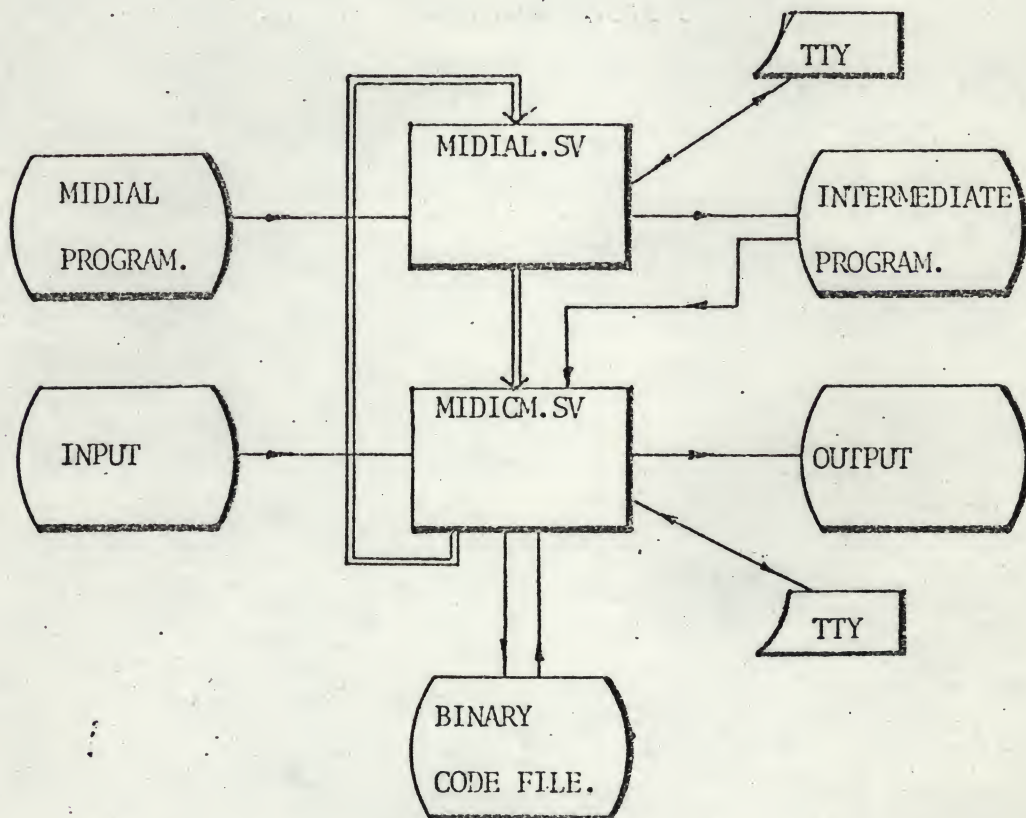
(MIDICM.SV)

The file command string is extended with an output file place for the preprocessor output, the intermediate program.

Two option switches are added also :

1. the /P -option switch, which specifies that only the preprocessing must be done. The system returns thereafter, by chaining to a new preprocessor system, with the asterisk of the Command Decoder.
2. the /L -option switch, which specifies that the intermediate program must be listed while compiling it with the Minial compiler.

If it is not used, only errorcodes without their corresponding line in error are visible on the teletype printer after error(s).



→ means chaining
to the program
save file

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS



6.2 METANOTIONS FOR THE MIDIAL SYNTAX.

ALPHA :: a;b;c;d;e;f;g;h;i;j;k;l;m;n;o;p;q;r;s;t;u;v;w;x;y;
 NOTION :: ALPHA ; NOTION ALPHA .

META PRODUCTION RULES.

EMPTY : .
 NOTION option : NOTION ; EMPTY .
 NOTION sequence : NOTION ; NOTION , NOTION sequence .
 NOTION list : NOTION ; NOTION , \odot , NOTION list .
 NOID : nonvoid ; void .
 Ω symbol : \odot .
 comma NOTION : comma , NOTION .
 NOTION comma : NOTION , comma .



PRODUCTION RULES.

letter	:	ALPHA symbol.
octal digit	:	(0); (1); (2); (3); (4); (5); (6); (7).
digit	:	octal digit ; (8); (9).
comma	:	(,).
quote	:	(").
quote image	:	quote , quote .
hek	:	(#).
escape	:	(%).
space	:	().
skip	:	(cr) symbol ; (lf) symbol ; (tab) symbol .
character	:	letter ; digit ; (&); ((); ()); ('); (:); (;); (/); (?); (+); (=); (.); (,); (-); (*); (!); ([); (]); (\$); (%); (\); (@); (^); (<); (>);
mark	:	character ; quote ; hek ; escape .
string token	:	character ; hek ; quote image ; escape ; space .
comment token	:	character ; quote ; escape ; space ; skip .
general character:	:	mark ; skip ; space .
octal integer	:	octal digit sequence .
number	:	digit sequence .
ld	:	letter ; digit .
identifier	:	letter ; letter , ld sequence option .
label identifier	:	identifier ; <u>G</u> , number ; <u>L</u> , number .
octal value	:	<u>B</u> , octal integer .
boolean value	:	<u>TRUE</u> ; <u>FALSE</u> .
character value	:	escape , mark ; <u>SP</u> ; <u>LN</u> .
string	:	quote , string token sequence option , quote .
comment	:	hek , comment token sequence option , hek .
open token	:	((); <u>IF</u> ; <u>CASE</u> ; <u>BEGIN</u> .
close token	:	() ; <u>FI</u> ; <u>ESAC</u> ; <u>OD</u> ; <u>END</u> .
exit	:	HALT .

THE UNIVERSITY OF CHICAGO

DEPARTMENT OF CHEMISTRY

1914

1915

1916

1917

1918

1919

1920

1921

1922

1923

1924

1925

1926

1927

1928

1929

1930

1931

1932

1933

1934

1935

1936

1937

1938

1939

1940

nonvoiding assignment operator :

$\odot \leftarrow$; $\odot =$; $\odot * \leftarrow$; $\odot =$; $\odot \leftarrow @$;
 $\odot = @$; $\odot * \leftarrow @$; $\odot = @$; operator and
 becomes .

operator and becomes :

dyac , $\odot \leftarrow$; dyac , $\odot =$.

nonvoiding monadic operator :

monop , $\odot @$ option ; A ; ST , $\odot @$;
 $\odot ..$, label identifier , $\odot ..$.

monop :

$\odot +$; $\odot -$; AP ; AR ; AW ; AY ; AZ ; C ;
M ; N ; NOT ; P ; R ; V ; W ; X ; Y ;
Z ; $\odot \$$; ALPHA ; DIGIT ; CHAR ; DECIN ;
DECS ; OCTIN ; OCTS ; COPY ; CALLPR ;
AX .

nonvoiding dyadic operator :

dyac , $\odot @$ option ; $\odot \uparrow$; COMP , $\odot @$;
EXPAND , $\odot @$; $\odot ..$, label identifier , $\odot ..$.

dyac :

$\odot +$; $\odot -$; $\odot *$; $\odot /$; $\odot >$; $\odot > =$; $\odot =$; $\odot / =$; $\odot .$;
 $\odot < =$; $\odot <$; $\odot \&$; $\odot !$; $\odot \leftarrow$; SL ; HD ; MEM .

nonvoiding proceduring operator :

SUBR ; EXPR ; PROG .

voiding monadic operator :

GOTO ; DCL .



label : label identifier , : .

constant : boolean value ; character value ; octal value ;
number ; label identifier .

vector : VEC , (, nonvoid quaternary list option ,) ;
[, nonvoid quaternary list option ,] ;
string .

call : label identifier , (, nonvoid quaternary list option ,) .

table item : label sequence option , constant ;
label sequence option , string .

nonvoid primary : constant ; vector ; closed nonvoid clause ;
identifier ; SREG .

nonvoid secondary: nonvoid primary ; call ;
nonvoiding monadic operator , nonvoid secondary ;
nonvoiding proceduring operator , NOID secondary .

nonvoid tertiary : nonvoid secondary ;
nonvoid tertiary , nonvoiding dyadic operator ,
nonvoid secondary .

nonvoid quaternary: nonvoid tertiary ; closed nonvoid quaternary train ;
nonvoid tertiary , nonvoiding assignment operator ,
nonvoid quaternary .

if chooser : nonvoid quaternary train , THEN , nonvoid quaternary train .

case prelude : open token , nonvoid quaternary train , IN .

out : OUT , nonvoid quaternary train .

nonvoid conditional: open token , if chooser, elif sequence option ,
ELSE , nonvoid quaternary train , close token ;
case prelude , nonvoid quaternary train list ,
out , close token .

elif : ELIF , if chooser .

void primary : exit ; while ; closed void clause ; RESET ; SKIP .

void secondary : void primary ; voiding monadic operator , nonvoid
secondary .

1. The first part of the document is a letter from the President of the United States to the Congress, dated January 3, 1862. It contains a report on the state of the Union and the progress of the war against the rebellion. The President mentions the recent victories of the Union forces and expresses confidence in the ultimate success of the cause.

2. The second part of the document is a report from the Secretary of the Treasury, dated January 10, 1862. It details the financial condition of the government and the measures taken to meet the demands of the war. The report notes the increase in public debt and the need for further financial support.

3. The third part of the document is a report from the Secretary of the Interior, dated January 15, 1862. It discusses the management of the public lands and the progress of the reclamation work. The report mentions the discovery of gold in California and the need for increased supervision of the mining industry.

4. The fourth part of the document is a report from the Secretary of the Navy, dated January 20, 1862. It describes the state of the naval forces and the progress of the construction of new ships. The report notes the success of the fleet in the Gulf of Mexico and the need for further expansion.

5. The fifth part of the document is a report from the Secretary of the War, dated January 25, 1862. It provides a detailed account of the military operations and the progress of the war. The report mentions the recent battles and the need for increased resources.

6. The sixth part of the document is a report from the Secretary of the State, dated February 1, 1862. It discusses the foreign relations of the United States and the progress of the diplomatic efforts. The report mentions the recent negotiations with Great Britain and the need for continued vigilance.

7. The seventh part of the document is a report from the Secretary of the Education, dated February 5, 1862. It describes the state of the public schools and the progress of the educational reforms. The report notes the success of the new curriculum and the need for further investment.

8. The eighth part of the document is a report from the Secretary of the Agriculture, dated February 10, 1862. It discusses the state of the agricultural industry and the progress of the reclamation work. The report mentions the success of the new farming techniques and the need for further support.

9. The ninth part of the document is a report from the Secretary of the Commerce, dated February 15, 1862. It describes the state of the commercial industry and the progress of the trade relations. The report notes the success of the new trade agreements and the need for continued vigilance.

10. The tenth part of the document is a report from the Secretary of the Finance, dated February 20, 1862. It discusses the state of the financial industry and the progress of the monetary reforms. The report mentions the success of the new currency and the need for further investment.

void tertiary : void secondary .

void quaternary : void tertiary ;

TABLE , open token , table item list , close token ;

CODE , open token , octal integer list , close token

NOID quaternary train: label sequence option , NOID quaternary ;

statement , NOID quaternary train .

balanced to void NOTION : NOID NOTION .

while : WHILE , nonvoid quaternary train , DO ,

balanced to void quaternary train , close token .

closed NOID quaternary train:

open token , NOID quaternary train , close token .

NOID quaternary trist:

nonvoid quaternary trist comma option , NOID quaternary

train , comma balanced to void quaternary trist option.

void conditional: open token , if chooservoid , elifvoid sequence option

else option , close token ;

case prelude , balanced to void quaternary trist ,

close token ;

case prelude , void quaternary trist , OUT , nonvoid

quaternary trist , close token .

if chooservoid : nonvoid quaternary train, THEN , balanced to void

quaternary train .

elifvoid : ELIF , if chooservoid .

else : ELSE , void quaternary train .

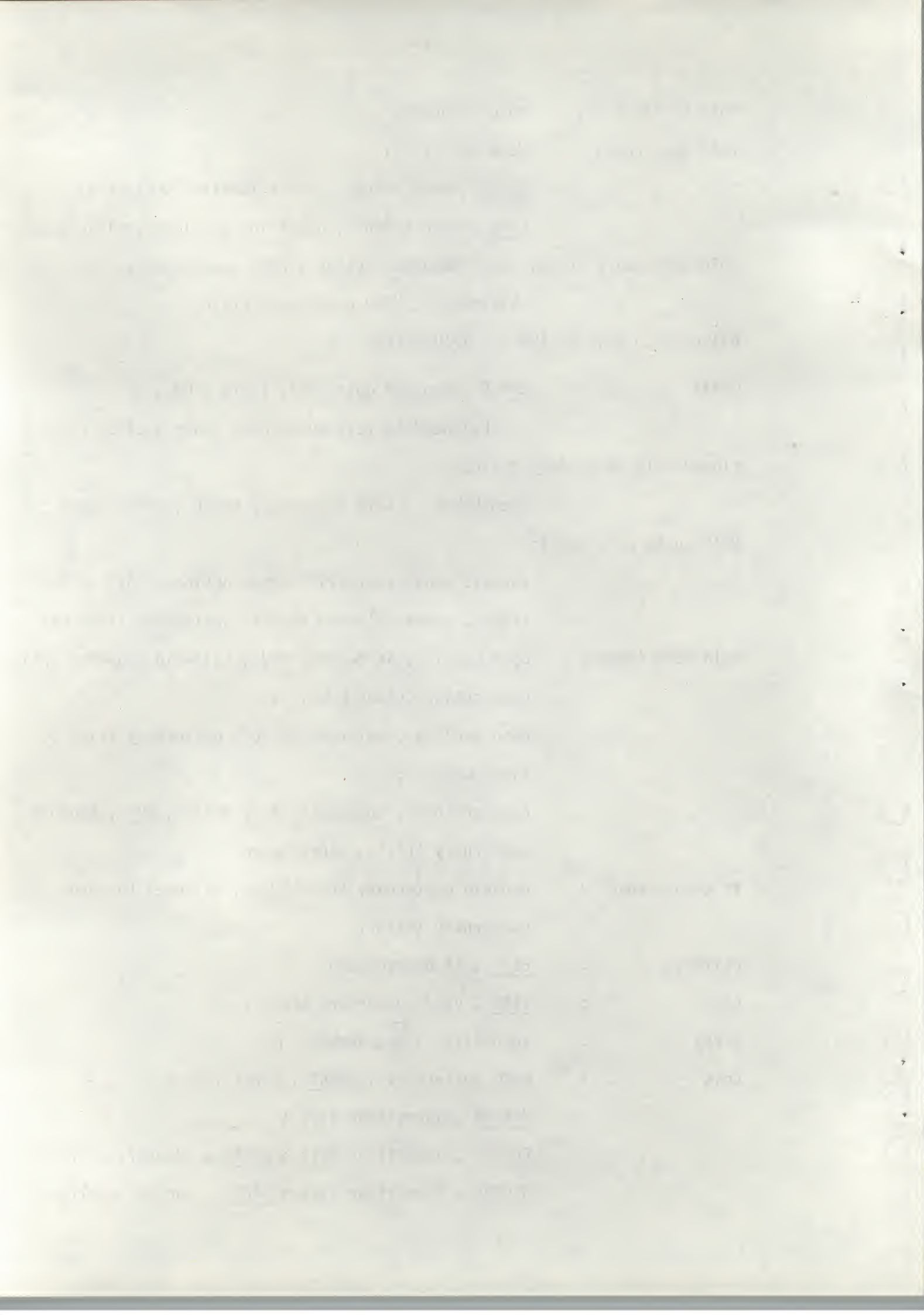
array : identifier , [, number ,] .

unit : NOID quaternary ; ARRAY , array list ;

VECTOR , identifier list ;

FORMAL , identifier list ; LOCAL , identifier list ;

GLOBAL , identifier list ; LABEL , identifier list .



closed NOID clause : NOID conditional ; closed NOID quaternary train .
sequencer : ; label sequence option .
statement : unit , sequencer .
program : label sequence option , unit .



6.3 THE SEMANTICS OF MIDIAL.

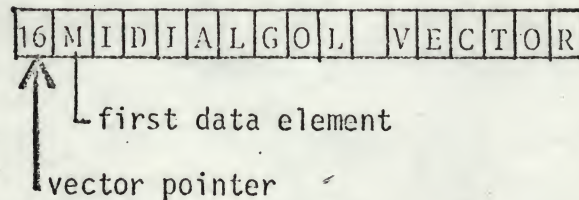
The framework of the MIDIAL language consists out of vectors.

Vectors are built up out of elements. The first element is the vector length, the other elements contain data.

The vector length gives the number of following data elements.

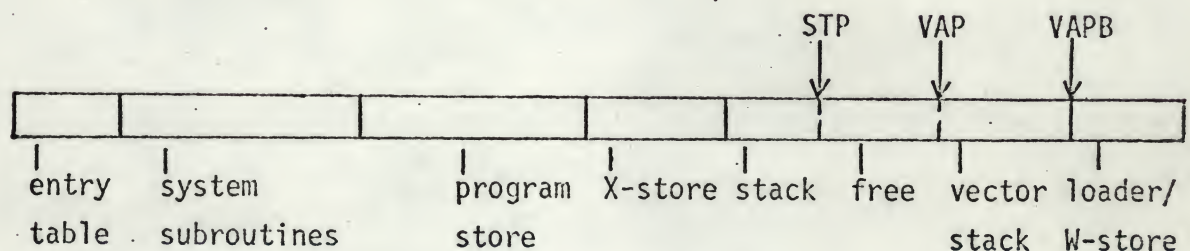
Vectors are located in consecutive memory locations. A reference to the vector is made by means of a pointer, which holds the address of the memory location which contains the vector length.

In the following figure a vector is drawn, which has a text as data.



The compiler and loaded programs are in the operating environment examples of vectors.

For a better notion of the MIDIAL semantics the store lay-out is treated :



- The entry table is a table of the entry points of the system routines.

These subroutines are available for the user by a coded indirect subroutine jump to this table. (JMS Z) This table is located in

Introduction

The purpose of this report is to provide a comprehensive overview of the current state of the art in the field of artificial intelligence. This report will discuss the various applications of artificial intelligence, the challenges associated with its development, and the future prospects of this technology. The report will also explore the ethical implications of artificial intelligence and the role of government in regulating its use.

Artificial Intelligence

Artificial intelligence (AI) is a branch of computer science that deals with the creation of intelligent machines that can perform tasks that would normally require human intelligence.

AI is a broad field that encompasses a wide range of sub-fields, including machine learning, natural language processing, and computer vision.

Machine learning is a sub-field of AI that deals with the development of algorithms that can learn from data and make predictions or decisions based on that data.



The machine learning pipeline is a process that involves training a model on a dataset, evaluating the model's performance, and then deploying the model to a production environment. The model is then monitored to ensure it continues to perform well over time.

page zero of the PDP-8 computer.

- A program is loaded in the program store by means of the system loader.

After the program a X-store is located. In this store values, created during the program elaboration, may be stored here.

- The stack is used by the system for holding intermediate results, return addresses, etcetera. The top of the stack is given by the stack top pointer (STP) .

- The vector stack stores vectors which are created during the evaluation of expressions.

The top of the vector stack is given by the vector stack pointer (VAP) . The bottom of the vector stack is given by VAPB .

At the call of a subroutine or a program (\$ or CALLPR) , on the stack top transferred parameters are placed, together with some information about the 'environment' (VAPB,AYO) , the VAPB pointer is set on the location pointed by VAP.

So every call evaluation is embedded within the free room of the outer environment. This principle is treated in detail by the procedure calls .

- The loader loads a program in the program store area, or in the X-store if the loader is called out of a program.

The memory locations which contains the loader, can be used as common variable room (or W-store) while destroying the loader.

variable and parameter stores.

The programmer may store a value, created during the elaboration of his program, into a memory location.

In principle all memory locations can be used for storing or modifying data, but there are three sorts of memory areas (stores) given by the system which are specifically for data storage or data transfer. These three areas are the W-store, or common variable room, the X-store, or global variable room, and the Y-store, or parameter and local variable room..

- the W-store , or common variable room :

This room can only be used if the loader is not needed by the programmer after the first access to the W-store.

The W-store overlays the system loader core locations, so the programmer must pay attention to this point.

The variables or constants stored in this W-store are accessible from every program. (see the PROG operator)

The store locations are numbered from 0 to 177_8 , they can be accessed by means of two operators, the AW operator and the W operator.

The AW operator working on the elaborated argument n (number value) gives the machine address of the n-th W-store location.

The W operator is the composition of the C operator (the contents of) and the AW operator : the operator applied to a number value m, refers to the contents of the m-th W-store location.

- the X-store , or global variable room :

All the variables stored in this area are accessible only from the program where the X-store belongs to.

Every called program has its own X-store, initially of 64 (100_8) X-store locations.

THE HISTORY OF THE

REIGN OF KING CHARLES THE FIRST

IN THE YEAR 1649

BY JOHN BURNET

OF THE UNIVERSITY OF OXFORD

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

LONDON, Printed by J. St. John, at the Black-Swan in St. Dunstons Church-yard, 1680.

IN TWO VOLUMES

For the main program, or executive, there are declared 100 X-stores by default, of which the X-store locations 0 to 20 are reserved for AX and X operations without restrictions.

If the programmer uses an identifier as variable, the identifier name refers to a store location in the X-store above the twentieth X-store location.

If the programmer needs more then the number of declared locations in the X-store, the programmer must declare them with the declare operator. The X-store can be shortened also by means of this declare operator if there are needed less X-store locations than the 100 default locations. (see DCL)

There are three operators defined for the access to the X-store locations: the AX operator applied to the number value m, gives the address of the m-th memory location. The A operator applied to an identifier gives the address of the X-store location where the identifier name refers to.

The C (contents of) operator applied to the A or AX operator gives the contents of the selected memory location.

For programmer convenience, the C operator followed by the AX operator may be replaced by the X operator. The C operator applied to an identifier preceded by the A operator, can be replaced by the identifier name only.

For assignation of a vector in the X-store, by means of an identifier where this vector is assigned to, the VECTOR or the ARRAY operator must be used. (see page 61)

This must be done for avoiding disturbances of the X-store locations which are referenced by means of identifiers and which are declared implicitly after the vector assignment.

The first part of the paper is devoted to a general discussion of the problem of the existence of solutions of the system of equations (1) for arbitrary values of the parameters $\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \iota, \kappa, \lambda, \mu, \nu, \xi, \omicron, \pi, \rho, \sigma, \tau, \upsilon, \phi, \chi, \psi, \omega, \varphi, \eta, \theta, \iota, \kappa, \lambda, \mu, \nu, \xi, \omicron, \pi, \rho, \sigma, \tau, \upsilon, \phi, \chi, \psi, \omega, \varphi$. It is shown that the system has solutions for arbitrary values of the parameters if and only if the following conditions are satisfied: $\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega + \varphi = 0$ and $\alpha^2 + \beta^2 + \gamma^2 + \delta^2 + \epsilon^2 + \zeta^2 + \eta^2 + \theta^2 + \iota^2 + \kappa^2 + \lambda^2 + \mu^2 + \nu^2 + \xi^2 + \omicron^2 + \pi^2 + \rho^2 + \sigma^2 + \tau^2 + \upsilon^2 + \phi^2 + \chi^2 + \psi^2 + \omega^2 + \varphi^2 = 0$. The second part of the paper is devoted to a detailed study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The third part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The fourth part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The fifth part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The sixth part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The seventh part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The eighth part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The ninth part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters. The tenth part of the paper is devoted to a study of the properties of the solutions of the system (1) for arbitrary values of the parameters. It is shown that the solutions of the system (1) for arbitrary values of the parameters are unique and depend continuously on the parameters.

- The Y-store, or parameter and local variable room :

This room can be accessed in two ways:

- a. By declaration of identifiers in this Y-store by means of the FORMAL operator for formal parameters, and the LOCAL operator for the local variables.
The identifiers following the FORMAL operator in the identifier list refer to the Y-store locations where, at the call of the subroutine, the actual values are filled in.

- b. access as numbered store locations for local usage or for parameter storing by means of the AY and the C AY (Y) operators.
If the user uses these Y-store locations for actual parameter transfer only, no declaration is needed for these locations. They are declared implicitly at the call of the subroutine.
The parameters are transferred as a vector, with in the location Y 0 the vector length. (The AY operator and the Y operator have the same meaning for the Y-store as the AX and the X operators for the X-store.)

This vector length gives the number of transferred parameters, so subroutines can be written for variable numbers of actual parameters.

If the programmer wants to use Y-store locations for local usage, they must be declared with the DCL operator.

DCL

This operator changes the number in the location AY 0 into the number which is the evaluation of the right-hand operand of this DCL operator.

In the main program the AY 0 pointer and the AX 0 pointer refer to the same address, so the DCL operator can also be used to declare X-store locations. (this means: change the Y-store vector length !)

THE HISTORY OF THE

REIGN OF

CHARLES THE FIRST

BY

JOHN

WILKINS

OF THE

UNIVERSITY OF

OXFORD

IN TWO VOLUMES

LONDON

PRINTED BY

JOHN WILKINS

AT THE

UNIVERSITY OF

OXFORD

IN TWO VOLUMES

LONDON

PRINTED BY

JOHN WILKINS

AT THE

UNIVERSITY OF

OXFORD

IN TWO VOLUMES

LONDON

PRINTED BY

If the programmer uses a variable number of parameters and some local variables (say 4), the local variables must be declared as follows :

SUBR(DCL(Y 0 + 4); .. and Y(Y 0 + 1) till Y(Y 0 + 4)
are the four local variables.

For further information about the parameter transfer, see the subroutine call and parameter handling on page 69 .

identifiers.

An identifier has no maximum length, but the first six identifier symbols only are significant for the recognition of the identifier by the preprocessor. Thus the first six identifier symbols of an identifier must be unique within the environment of the program part where this identifier is used.

Of course local identifiers, declared with the FORMAL, LOCAL or LABEL declarational operators may have the same identifier name as a global defined identifier.

Depending on the context identifiers, which are not defined already local, are defined automatically global, either as X-store variable, or as G(-lobal) label. Sometimes identifiers are defined implicitly, depending on the context, even if they were declared already in another mode. The programmer must pay attention to this.

The rules for this implicit definition are :

1. After a V operator or a GOTO operator or before a colon (:), a label identifier is expected. Thus even if the identifier is defined already as a X-store variable, there is made a global label with the same name.

To avoid this situation, the identifier name must be embedded

THEORY

The first part of the theory is the definition of the function $f(x)$ and the function $g(x)$. The second part is the definition of the function $h(x)$ and the function $k(x)$. The third part is the definition of the function $l(x)$ and the function $m(x)$. The fourth part is the definition of the function $n(x)$ and the function $o(x)$. The fifth part is the definition of the function $p(x)$ and the function $q(x)$. The sixth part is the definition of the function $r(x)$ and the function $s(x)$. The seventh part is the definition of the function $t(x)$ and the function $u(x)$. The eighth part is the definition of the function $v(x)$ and the function $w(x)$. The ninth part is the definition of the function $x(x)$ and the function $y(x)$. The tenth part is the definition of the function $z(x)$ and the function $z(x)$.

The first part of the theory is the definition of the function $f(x)$ and the function $g(x)$. The second part is the definition of the function $h(x)$ and the function $k(x)$. The third part is the definition of the function $l(x)$ and the function $m(x)$. The fourth part is the definition of the function $n(x)$ and the function $o(x)$. The fifth part is the definition of the function $p(x)$ and the function $q(x)$. The sixth part is the definition of the function $r(x)$ and the function $s(x)$. The seventh part is the definition of the function $t(x)$ and the function $u(x)$. The eighth part is the definition of the function $v(x)$ and the function $w(x)$. The ninth part is the definition of the function $x(x)$ and the function $y(x)$. The tenth part is the definition of the function $z(x)$ and the function $z(x)$.

between parenthesis as a closed nonvoid quaternary.

Then the whole identifier name list is searched through.

So the MINIAL program :

G1: X1:=G1;....; IF (X2-:=1) /= 0 THEN GOTO X1 FI;....

can be written in MPDIAL as :

LAB: X:=LAB;....;IF (CNT-:=1) /= 0 THEN GOTO (X) FI;... .

2. In front of an assignation operator (\leftarrow or $:=$) an X- or Y- store identifier is expected .
3. The identifier name of a call must always be a label, and must always be an identifier. So if the identifier was not defined at the moment of the call as a label, a global (G) label is generated.

If an identifier is found, in the preprocessor phase, which is not context dependent, a search is made in the identifier list.

The sequence of searching an identifier is :

1. search through all local labels (L).
2. search through all local variables (Y).
3. search through all global labels (G).
4. search through all global variables (X).

if the identifier is not found, it is set into the identifier list as a X-store variable.

If an identifier is context dependent, a search is made or through all Y and X-store names, or through all L and G-store names, in that order. If the identifier is not found in the list, it is set in the identifier list as a global name, or X or G.

THE UNIVERSITY OF CHICAGO

THE DIVISION OF THE PHYSICAL SCIENCES

DEPARTMENT OF CHEMISTRY

RECEIVED JANUARY 10, 1955

BY THE DEPARTMENT OF CHEMISTRY

FROM THE DEPARTMENT OF CHEMISTRY

OF THE UNIVERSITY OF CHICAGO

CHICAGO, ILLINOIS

TO THE DEPARTMENT OF CHEMISTRY

OF THE UNIVERSITY OF CHICAGO

CHICAGO, ILLINOIS

U.S.A.

THE UNIVERSITY OF CHICAGO

THE DIVISION OF THE PHYSICAL SCIENCES

DEPARTMENT OF CHEMISTRY

RECEIVED JANUARY 10, 1955

BY THE DEPARTMENT OF CHEMISTRY

FROM THE DEPARTMENT OF CHEMISTRY

OF THE UNIVERSITY OF CHICAGO

CHICAGO, ILLINOIS

TO THE DEPARTMENT OF CHEMISTRY

OF THE UNIVERSITY OF CHICAGO

CHICAGO, ILLINOIS

U.S.A.

THE UNIVERSITY OF CHICAGO

THE DIVISION OF THE PHYSICAL SCIENCES

DEPARTMENT OF CHEMISTRY

labels.

There are two sorts of label: local labels and global labels. Local labels are defined only with SUBR, EXPR and PROG texts. The compiler deletes all local label definitions from its label list on reaching the end of the procedure text in which the label is defined.

Local label can be defined in two ways:

1. by means of a declaration after the proceduring operator (after PROG,SUBR,EXPR). This declaration is done with the declarational operator LABEL followed by a list of identifiers. The preprocessor generates of these identifiers L-labels.

L-labels are labels consisting out of the basic symbol L, followed by a number. The preprocessor generates numbers above twenty.

2. by means of L-labels directly, but only for the numbers one to twenty.

If the first option is not used, it is allowed to use numbers above twenty.

Note: in another proceduring environment on the same level, these labels can be used again because of their locality.

Global labels are defined everywhere in a program. They are local within every PROG proceduring environment.

Global label identifiers can be given in two ways:

1. as identifiers, they are converted to G-labels by the preprocessor. G-labels consists of the basic symbol G, followed by a number.

If it is not clear to the preprocessor that a global label is used, then the global label identifier must be declared with the GLOBAL operator in its following label.

For example the MINIAL program part X1:G1;G1: must be written in MIDIAL as: 'GLOBAL G ; X:= G ; G :

Notice that if the label identifier occurs in the program for the first time in a context in which it is defined implicitly, the GLOBAL operator is not needed.

2. as G-labels: by means of the basic symbol G, followed by a number. If the first option, an identifier as global label, is used, only a number between one and twenty one can be used. Otherwise all possible numbers can be used.

A label is an label identifier followed by a colon. A label points to the first instruction of the quaternary with bears that label.

The label value gives the machine address at run time of the location which holds the first instruction of the quaternary pointed to by that label.

comment.

Comments may occur between any pair of MIDIAL symbols, except within the character sequence of basic symbols or identifiers. All information of a comment, including the two hek symbols are disregarded by the compiler.

In all the declaring operator lists (GLOBAL, LABEL, FORMAL, LOCAL, ARRAY and VECTOR) comments are not allowed before the semicolon. (;)

1870

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

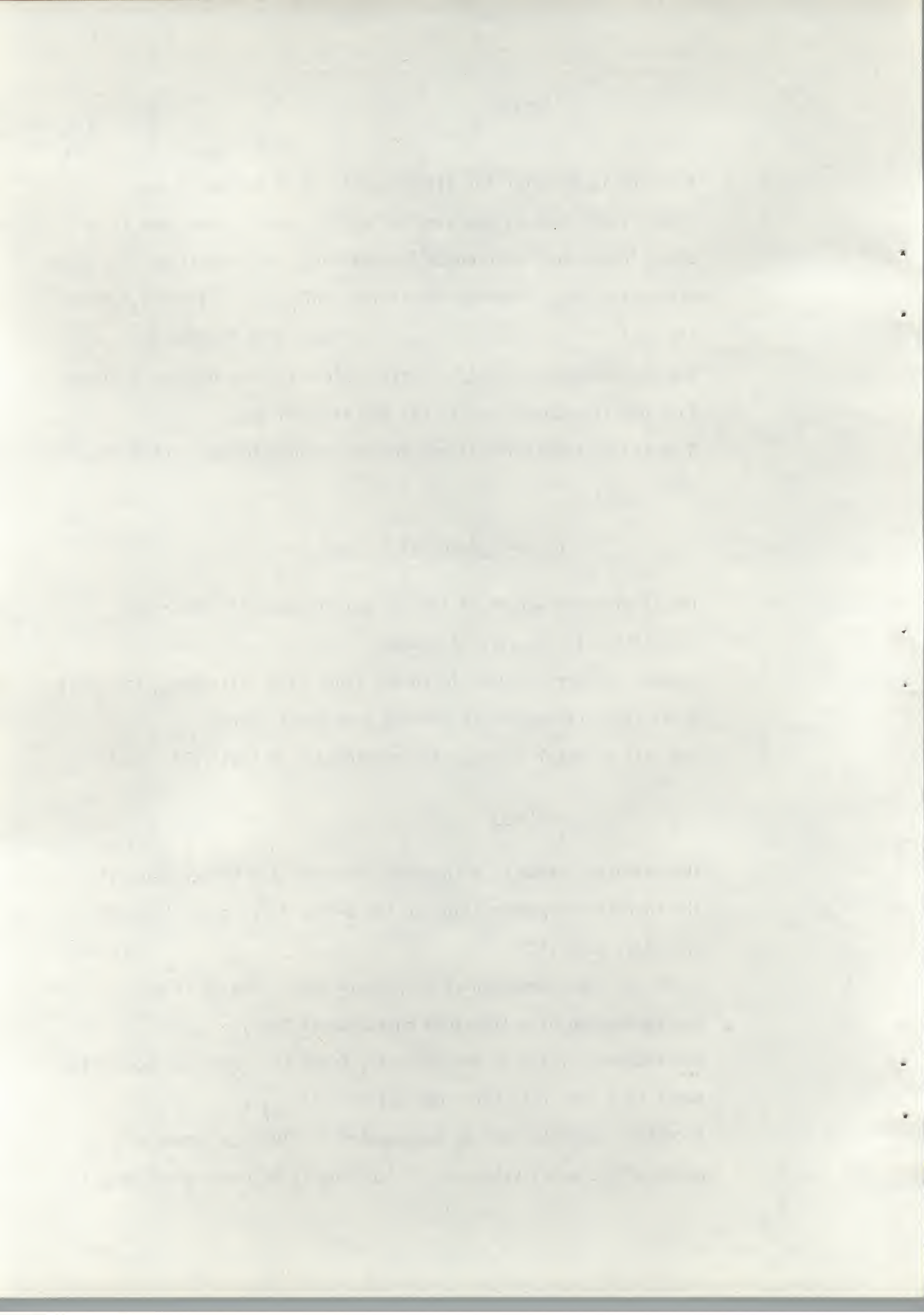
...

...

...

...

A control character can be represented in MIDIAL programs only by means of the octal value. So $\uparrow Z$ (control Z) is represented by B 232 .



The octal value of an octal integer is represented by a basic symbol B, followed by that octal integer. It represents the octal value of that integer, which can be used for calculation within programs.

The value of the boolean TRUE is -1 (decimal) or B 7777, and the value of the boolean FALSE is 0 or B 0.

The logical value TRUE or FALSE in calculations without comparing operators is negative, respectively positive, e.g. the conditional

IF (X:=7; X+Y) THEN a ELSE b FI is elaborated as follows:
if Y is greater than seven the else-clause b is taken, otherwise the then-clause a is executed.

algebraical operators

All the algebraical operators operate on full bit operands.

There are two monadic operators : the + operator and the - operator.

The monadic - operator takes the complement value of the operand seen as number. In MIDIAL the two's-complement is used for the addition and the subtraction of numbers. This means that the negative value of a number is equal to the complement of that number (all bits inverted incremented by one).

For -5 this is done as following:

	000 000 000 101 ₂	or 5
one's complement :	111 111 111 010 ₂	
increment :	111 111 111 011 ₂	or -5 or <u>B 7773</u> .

Notice that the left most bit can be regarded as sign bit.

There are four dyadic operators for calculation : the + (add), the - (subtract) , the * (multiply) and the / (divide) operator.

The first of these is the fact that the
the second is the fact that the
the third is the fact that the
the fourth is the fact that the
the fifth is the fact that the
the sixth is the fact that the
the seventh is the fact that the
the eighth is the fact that the
the ninth is the fact that the
the tenth is the fact that the

The first of these is the fact that the
the second is the fact that the
the third is the fact that the
the fourth is the fact that the
the fifth is the fact that the
the sixth is the fact that the
the seventh is the fact that the
the eighth is the fact that the
the ninth is the fact that the
the tenth is the fact that the

The first of these is the fact that the
the second is the fact that the
the third is the fact that the
the fourth is the fact that the
the fifth is the fact that the
the sixth is the fact that the
the seventh is the fact that the
the eighth is the fact that the
the ninth is the fact that the
the tenth is the fact that the

There are no priorities for dyadic operators, this applies also to all other dyadic operators, listed under dyacs in the syntax. They are elaborated from left to right. E.g. $X + X + Y * Z$ has the same action as $(X + Y) * Z$.

logical operators.

These operators perform action on the bits of the operand(s) itself. The monadic NOT operator performs inversion of its operand bit for bit. This has the effect that NOT N is converted to $-N-1$.

There are three dyadic operators : the & (and), the ! (or) and the \leftarrow (shift left) operators . For the and-operator and the or-operator the result per bit is depending on the bit values of its operands.

A table of results for all possibilities is listed below:

bit i operand 1	bit i operand 2	&	!
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

If the reader replaces the zeroes by false and the ones by true the table can be used as result table for boolean operands.

The operand at the right hand side of the shift left operator defines the number of shifted places of the bits of the operand at the left hand side of the shift operator.

Let x_1, x_2, \dots, x_n be the components of the vector x in the basis $\{e_1, e_2, \dots, e_n\}$. Then the vector x can be written as

$$x = x_1 e_1 + x_2 e_2 + \dots + x_n e_n.$$

Let y_1, y_2, \dots, y_m be the components of the vector y in the basis $\{f_1, f_2, \dots, f_m\}$. Then the vector y can be written as

	x_1	x_2	x_3	x_4
y_1	1	0	0	0
y_2	0	1	0	0
y_3	0	0	1	0
y_4	0	0	0	1

Let z_1, z_2, \dots, z_k be the components of the vector z in the basis $\{g_1, g_2, \dots, g_k\}$. Then the vector z can be written as

A negative shift number defines a shift to right over the number of bit places given by the positive number, e.g. $\underline{B} 0025 \leftarrow 2$ is equal to $\underline{B} 0124$ and $\underline{B} 0025 \leftarrow -2$ is equal to 5.

At shifting left, the right most bit position is filled with a bit zero, at shifting right, the left bit is filled with bit zero.

compare operators

All the compare operators are automatically dyadic and return with a boolean value depending on the two operands.

The operators are : the < operator, or less than operator;
the <= operator, or less than or equal operator;
the = operator, or equal to operator;
the /= operator, or not equal operator;
the > operator, or greater than operator;
and the >= operator, or greater than or equal operator.

If the equation holds the boolean value true is generated as the result of it, otherwise the boolean value false is given as result.

assignation.

A value assigned to a memory location by means of the dyadic operator $:=$. This operator takes an address (left hand operand) and a value (right hand operand), and assigns the value to the address. This means that the value is placed in that location.

The nonvoiding value of the operand is the value which has been assigned, so that chain assignation is possible.

Thus $\underline{AX} 3 := \underline{A} \text{ STORE} := 6$; assigns 6 to the location which has the name STORE and also to the location on the third address in the X-store.



For programmer convenience the := operator may be used in stead of the :- operator provided that the operand at the left hand side of it is, or is converted into, a secondary of address mode and is preceded by a \underline{C} operator. The compositions with the \underline{C} operator as \underline{X} , \underline{W} , \underline{Y} and identifiers which represent variables or parameters are allowed too.

Thus the assignment of the example above can also be written as
 $\underline{X} \text{ 3} := \text{STORE} := 6 ;$

These conventions mean that identifiers may be regarded as variables as in ALGOL. However the distinction between the address of a value and the value itself is maintained strictly. This distinction bears fruit when passing parameters to subroutines, which has the same generality in MIDIAL as in ALGOL 68.

The two vector assignation operators (*+ and *=) are treated in the chapter vector operations (see page 53). For the form *= the same conventions apply as for the := operator compared with the :- operator.

Operator and becomes.

The action involved in elaborating $A \text{ } \Omega \text{ } \oplus \text{ } V$, where A is an address, V is a value and Ω is a dyac. (dyadic operator), is the same as the action involved by $A \text{ } \oplus \text{ } \underline{C} \text{ } A \text{ } V$. This with the exception of side effects in the evaluation of the address part A .

The elaboration of $A \text{ } \Omega \text{ } \ominus \text{ } V$ is equal to $A \text{ } \ominus \text{ } A \text{ } \Omega \text{ } V$, where A is a reference the the value at the address \underline{A} .

For example : the elaboration of $\text{CNT} \text{ } \oplus \text{ } := 1 ;$ is an incrementation of the value of CNT . And the elaboration of $\text{X} \text{ } \leq \text{ } := 5 ;$ is equal to $\text{X} := \text{IF } \text{X} \leq 5 \text{ THEN TRUE ELSE FALSE FI} ;$.

The first part of the paper is devoted to a general discussion of the problem. It is shown that the problem is equivalent to the problem of finding a function $f(x)$ which satisfies the conditions

$$f(x) = \int_0^x f(t) dt + \int_0^x f(t) dt + \dots$$

It is shown that the problem is equivalent to the problem of finding a function $f(x)$ which satisfies the conditions

$$f(x) = \int_0^x f(t) dt + \int_0^x f(t) dt + \dots$$

It is shown that the problem is equivalent to the problem of finding a function $f(x)$ which satisfies the conditions

$$f(x) = \int_0^x f(t) dt + \int_0^x f(t) dt + \dots$$

It is shown that the problem is equivalent to the problem of finding a function $f(x)$ which satisfies the conditions

$$f(x) = \int_0^x f(t) dt + \int_0^x f(t) dt + \dots$$

vectors.

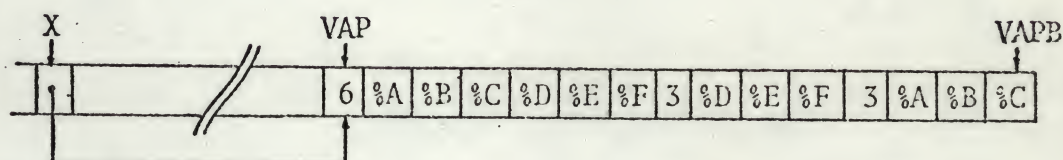
A vector consists out of a number of elements. These elements are located in consecutive memory locations preceded by the length of the vector. A vector whose elements are character values may be written as a string, e.g. VEC(%E,%,%L,SP) is equivalent to the string "E.L " .

A null vector is a vector with the length zero. It is written as VEC() or "" or [] .

The value of a vector (the object required by all vector operations) is the address of the location which contains the vector length. This value can be treated as a vector pointer if it is saved in a memory location. The elaboration of a vector causes it to be created in the vector area at the top of the vector stack. (VAP)

This vector stack top moves towards the stack top (STP), so the vectors are created top down in the core.

The elaboration of `X := "ABC"."DEF" ;` results in a vector stack as:



The length and the elements of a vector may be copied into consecutive memory locations by the use of the vector assignation operators.

These operators are the `*←` operator and the `*=` operator.

The two assignation operators `:=` and `:=` only perform the transfer of the vector pointer, this as distinct from the vector assignation operators which copy the vector to its destination and reset the vector area pointer (VAP) to its initial value of the vector area bottom pointer (VAPB) of its own proceduring environment.

This last action has the effect of a garbage collection: every new

created vector overwrites previous defined vectors.

RESET

This garbage collection is explicitly done by the evaluation of the RESET instruction. The action of the RESET operator can be written in MIDIAL as : EXPR(RESET: VAP:=VAPB); .

This operator is used for garbage collection in programs which use for assignments of vectors the :← and the := assignation operators only. These operators do not influence the vector stack.

vector operators.

The vector operators take one (monadic) or two (dyadic) vectors as their operands, or better : operate with vector address pointers pointing to vectors.

COPY

The monadic vector operator COPY makes a copy vector of its vector operand in the vector stack and returns with the vector address of this copy vector.

.

The . operator or concatenation operator joins two vectors.

For example: the elaboration of "SWITCH" . "BOARD" creates the vector "SWITCHBOARD" and returns with the address of this vector.

SL

The SL operator or slice operator takes two vectors as its operands and slices the first according to the first two elements given in its second operand. If the second number is less than the first number a null vector is returned as result of the operation.

Slicing with a vector of one element n (>0) or with a null vector gives unpredictable results.

Thus the elaboration of "FATHER" SL VEC(3,5) has as result the vector "THE" ; the elaboration of "COMPETITION" SL VEC(3,4,7,9) has the value "MP" and the elaboration of VEC(1,2,3,7,8) SL VEC(3) has as result the vector VEC(3,7,8) .

The first of these is the fact that the
1918-1919 season was a very dry one
and the crops were much smaller than
in previous years. This was due to the
fact that the rainfall was much less
than in previous years. The second
fact is that the crops were much
smaller than in previous years. This
was due to the fact that the rainfall
was much less than in previous years.

1918-1919

The third fact is that the crops were
much smaller than in previous years.
This was due to the fact that the
rainfall was much less than in
previous years. The fourth fact is
that the crops were much smaller
than in previous years. This was
due to the fact that the rainfall
was much less than in previous years.

1919-1920

The fifth fact is that the crops were
much smaller than in previous years.
This was due to the fact that the
rainfall was much less than in
previous years. The sixth fact is
that the crops were much smaller
than in previous years. This was
due to the fact that the rainfall
was much less than in previous years.

1920-1921

The seventh fact is that the crops were
much smaller than in previous years.
This was due to the fact that the
rainfall was much less than in
previous years. The eighth fact is
that the crops were much smaller
than in previous years. This was
due to the fact that the rainfall
was much less than in previous years.

MEM The MEM operator or member of-operator searches in the second operand for occurrences of the first operand as a substring.

A vector containing all the start positions of these occurrences in the second operand. For example: the elaboration of "EN" MEM "INDEPENDENT" delivers the vector VEC(6,9) :

HD The HD operator or head of-operator returns the boolean value TRUE if the first operand is a heading substring of the second vector. If the first operand is not a heading substring of the second vector operand, the boolean value FALSE is returned. For example the elaboration of VEC(2,4) HD VEC(2,4,7,%A) delivers the value TRUE , and the elaboration of "AJAX" HD "AC MILAN" has the value FALSE .

ⓐ In addition to these operators, nearly all operators, algebraical as well as vector operators, have a corresponding vector operator which takes a vector, or a pair of vectors, as operand(s) and perform the operation given by the operator element by element. They are distinguished by being followed by the @ symbol.

The dyac Ω followed by the @ operator (at-operator) has the following capabilities :

1. the operand vectors are of equal length :

the dyac is applied element by element on the two vectors.

For example: the elaboration of VEC(a,b,c) Ω @ VEC(p,q,r) results in the vector VEC(a Ω p, b Ω q, c Ω r) .

2. the first vector operand is longer than the second vector operand:

the operations are done only for the elements which the second possess . For example: the result vector of the program part

VEC(a,b,c,d) Ω @VEC(p,q) is the vector VEC(a Ω p, b Ω q) .

3. if the second vector operand is the null vector, then the result of the operation is a null vector, e.g. the elaboration of $\text{VEC}(a,b) \ \Omega \ @ \ \text{VEC}()$ delivers the null vector $\text{VEC}()$.
4. if the first operand is the null vector, then the result vector is an undefined vector of the length of the second operand vector. For example: the value of $\text{VEC}() \ \Omega \ @ \ \text{VEC}(a,b,c,d)$ is an undefined vector of length four .
5. if the first vector operand has a smaller length than the second one, but none of them is the null vector :
 - for the corresponding elements is the operation done as for vectors of equal length, but for the last element of the first operand, the operation is repeated with all the remaining elements of the second operand by operating with the intermediate result.
 For example the elaboration of the program part
 $\text{VEC}(a,b,c) \ \Omega \ @ \ \text{VEC}(p,q,r,s,t)$ has the vector
 $\text{VEC}(a \ \Omega \ p, b \ \Omega \ q, (((c \ \Omega \ r) \ \Omega \ s) \ \Omega \ t))$ as result .

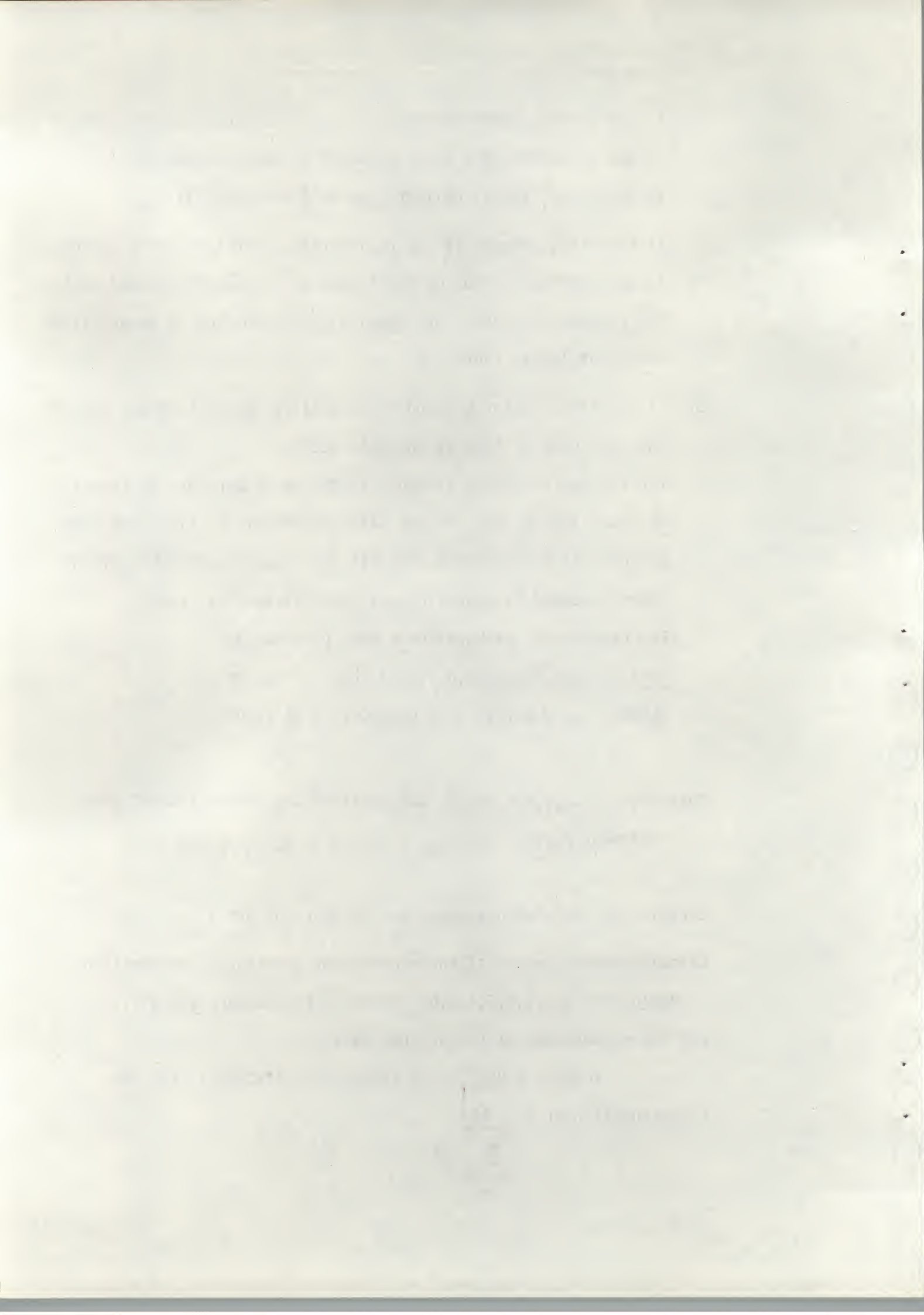
Note: a,b,c,p,q,r,s,t denote values after the elaboration of that element part.

Examples of calculations using the @ operator are :

Calculation of the sum of the elements of a vector : elaboration of $\text{VEC}(0) + @ \text{VEC}(1,2,3,4,5,6)$ delivers the vector $\text{VEC}(21)$, and the elaboration of the program part

: $(\text{CNT}:=0 ; \text{VEC}(0) + @ (10+((\text{CNT}+:=1)*\text{CNT})))$ is the representation of :

$$\sum_{i=1}^{10} i^2$$



(the \dagger operator elaborates its second operand the number of times denoted by the first operand, and deposits the intermediate results of each elaboration of the second operand in a vector.)

Cyclic interchange can be done by the following program part:

C @ VEC(A H,A I,A J) := @ VEC(I,J,H); .

There are still three vector operators undefined :

COMP@

The COMP basic symbol followed by the @ operator compress its first vector operand with the second operand as follows :

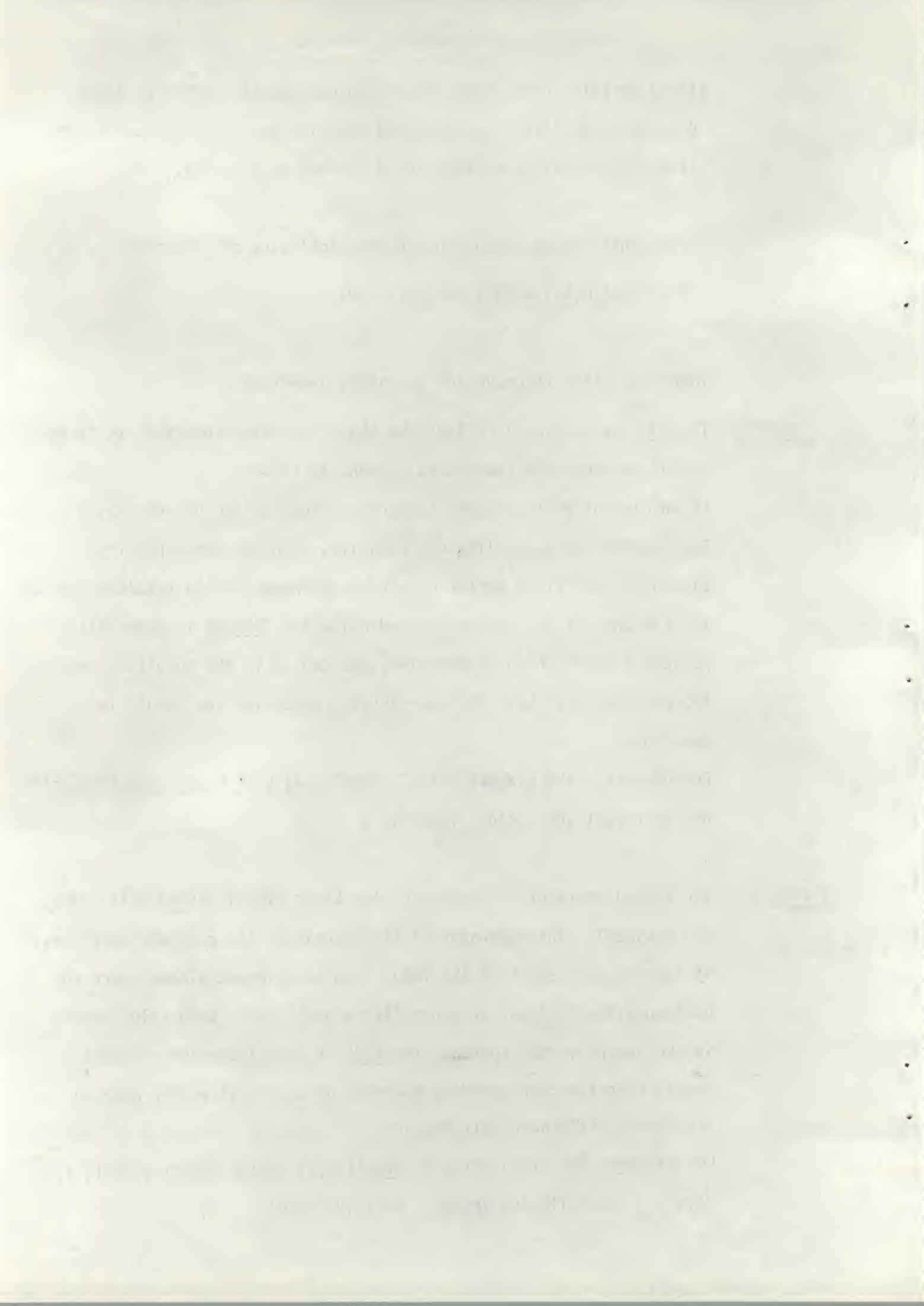
if an element of the second operand is equal to the boolean value FALSE ,or is in generality not negative, then the corresponding element of the first vector operand is dropped for the resulting vector. The elements of the second operand which are TRUE,or in generality possess a value which is negative, are copied in the resulting vector. The operands must have the same length, otherwise the result is undefined.

For example : the elaboration of VEC(3,5,4) COMP @ VEC(TRUE,FALSE,-10) has as result the vector VEC(3,4) .

EXPAND @

The expand operator is denoted by the basic symbol EXPAND followed by the @ symbol . This operator is the opposit of the compress operator, it inserts an element of the value 0 on the element places where the boolean value FALSE or in generality a positive or zero value occurs in the second vector operand. The TRUE or negative value elements denote that the corresponding elements of the first vector operand are copied into the result vector.

For example: the elaboration of VEC(1,2,3) EXPAND @ VEC(-1,0,-1,-1,0) delivers the following vector VEC(1,0,2,3,0) .



And the elaboration of the program part

$\text{MASK} := \text{VEC}(-1,0,0,-1,0,-1)$; $\text{TXT1} := \text{"ABC"}$; $\text{TXT2} := \text{"DEF"}$;

(TXT1 EXPAND @ MASK) + @ (TXT2 EXPAND @ NOT @ MASK) has the
between result $\text{VEC}(A,0,0,B,C,0)$ + @ $\text{VEC}(0,D,E,0,0,F)$ and as result
the vector $\text{VEC}(A,D,E,B,C,F)$.

ST@ The basic symbol ST followed by the @ operator is called the straighten operator. It flatten a vector of vectors.

The elaboration of ST @ VEC("AB",VEC(1,4,5),OCTS(B 7654))
delivers the vector VEC(%A,%B,1,4,5,%7,%6,%5,%4) .

This operator is used often to print the decimal conversion of all
vector elements of a vector :

ST @ DECS @ VEC(.....) . (see the example of the triangle
of Pascal).

closed clauses.

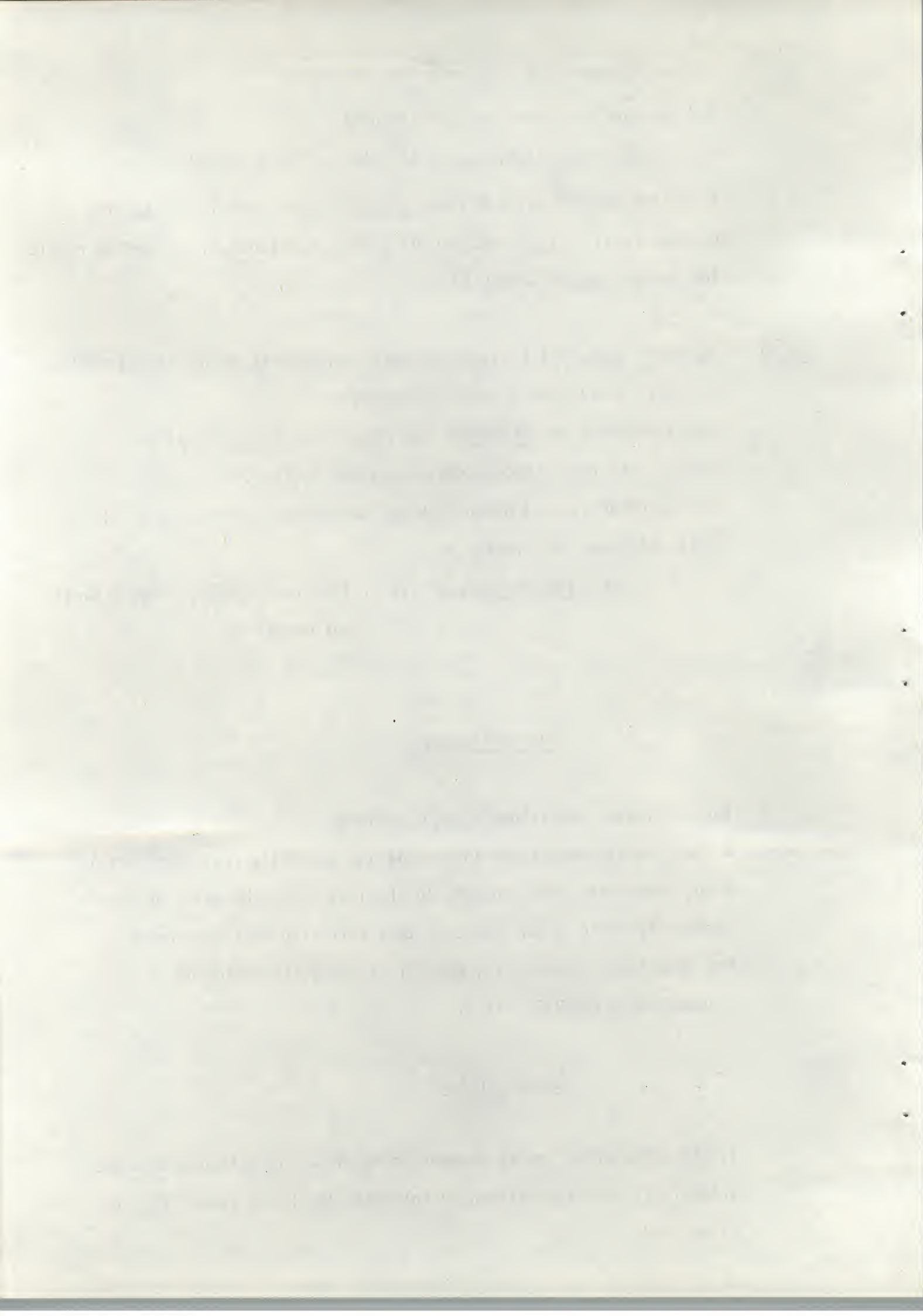
Closed clauses are either void or nonvoid.

A closed quaternary train is void if its textually last quaternary is
void, otherwise it is nonvoid. In the last case, the value of the
quaternary train is the value of that textually last quaternary .

For example : ($\text{COUNT} := 1$; RESET) is void, the value of
($\text{COUNT} := 0$; $\text{COUNT} + 7$) is 7.

conditionals.

If the selector of the if chooser is negative, in particular has the
value TRUE, then the quaternary following the basic symbol THEN is
elaborated.



Otherwise the elif sequence is elaborated if this sequence is specified, else the quaternary train following the basic symbol ELSE is elaborated, if it is specified.

If the else-clause does not exist, the conditional is balanced to void.

The basic symbol ELIF is a combination of ELSE and IF, but the if of it has no opening function as the basic symbol IF has.

This means that one closing symbol closes a IF..THEN..ELIF..THEN..ELIF..THEN...ELSE.. construction. The preprocessor inserts before this closing symbol a number of closing symbols for the elifs of the inner conditionals.

For example : IF X=%(THEN C+=1;"('(' ELIF X=%' THEN "(:(' ELSE VEC(X) FI delivers for a (symbol as value of X a vector with a (symbol, a ' symbol and a (symbol. Meanwhile C is incremented with 1. If X has not the character value of an open parenthesis or of a single quote sign, the value of the operation is a vector with as element the value of X.

The selector of a case-conditional with value n selects the n-th quaternary after the basic symbol IN. If its value n is less than one or greater than the number of elements in the quaternary train list after the IN, then the quaternary train after the basic symbol OUT is selected, if it is selected.

If the out-case is not specified, the selectands are balanced to void.

For example : V CASE PARAMETER IN LAB1,LAB2,LAB3 OUT LAB4 ESAC ; selects the calling of an expression depending on the value of PARAMETER. If PARAMETER has the value 2, expression LAB2 is chosen, if it has a value less than 1 or greater than 3, the expression LAB4 is implemented.

1. The first part of the paper is devoted to a general discussion of the problem.

2. In the second part, we shall consider the case of a single particle.

3. The third part is devoted to the case of a system of particles.

4. In the fourth part, we shall consider the case of a continuous medium.

5. The fifth part is devoted to the case of a system of continuous media.

6. In the sixth part, we shall consider the case of a system of particles and continuous media.

7. The seventh part is devoted to the case of a system of particles and continuous media.

8. In the eighth part, we shall consider the case of a system of particles and continuous media.

9. The ninth part is devoted to the case of a system of particles and continuous media.

10. In the tenth part, we shall consider the case of a system of particles and continuous media.

11. The eleventh part is devoted to the case of a system of particles and continuous media.

12. In the twelfth part, we shall consider the case of a system of particles and continuous media.

13. The thirteenth part is devoted to the case of a system of particles and continuous media.

14. In the fourteenth part, we shall consider the case of a system of particles and continuous media.

15. The fifteenth part is devoted to the case of a system of particles and continuous media.

16. In the sixteenth part, we shall consider the case of a system of particles and continuous media.

17. The seventeenth part is devoted to the case of a system of particles and continuous media.

18. In the eighteenth part, we shall consider the case of a system of particles and continuous media.

19. The nineteenth part is devoted to the case of a system of particles and continuous media.

20. In the twentieth part, we shall consider the case of a system of particles and continuous media.

21. The twenty-first part is devoted to the case of a system of particles and continuous media.

22. In the twenty-second part, we shall consider the case of a system of particles and continuous media.

23. The twenty-third part is devoted to the case of a system of particles and continuous media.

24. In the twenty-fourth part, we shall consider the case of a system of particles and continuous media.

25. The twenty-fifth part is devoted to the case of a system of particles and continuous media.

26. In the twenty-sixth part, we shall consider the case of a system of particles and continuous media.

27. The twenty-seventh part is devoted to the case of a system of particles and continuous media.

28. In the twenty-eighth part, we shall consider the case of a system of particles and continuous media.

29. The twenty-ninth part is devoted to the case of a system of particles and continuous media.

30. In the thirtieth part, we shall consider the case of a system of particles and continuous media.

And the program statement C(PAR THEN A CNT1 ELSE A CNT2) --:= 25 ;
is an conditional assignation. Here the advantage of the distinction
between an address and a value is clear.

while.

The while construction WHILE condition DO body OD ; has the
effect of a repetition of the elaboration of the body part until
the condition becomes a value which is not negative, particularly
becomes the boolean value FALSE .

The while construction WHILE TRUE DO....OD; defines an infinite
program loop, and this loop can only be interrupted above the
program elaboration level, e.g. by striking the +C (control C)
key at the system keyboard.

create.

The + operator or create operator procedures its second operand
into an expression (see expressions, page 67) , calls it the
number of times given by the operand at the left side of the +
operator, and constructs a vector of the values of the called
expression .

By voiding the result vector, this operator can be seen as a
for-loop of ALGOL .

Thus the value of 3 +(2 +5) VEC(VEC(5,5), VEC(5,5), VEC(5,5))
and the statement STORE := 50 +0 is a way to set the 50 store
locations following STORE to zero.

1861

The first of the year was a very dry one, and the crops were much injured. The weather was very hot, and the ground was very dry. The crops were much injured, and the yield was very small. The weather was very hot, and the ground was very dry. The crops were much injured, and the yield was very small.

1862

The second of the year was a very wet one, and the crops were much injured. The weather was very cold, and the ground was very wet. The crops were much injured, and the yield was very small. The weather was very cold, and the ground was very wet. The crops were much injured, and the yield was very small.

declarational operators.

The VECTOR operator reserves store locations in the X-store for vector operations with identifiers into this store.

The identifiers in the identifier list are allocated consecutively to memory locations in the X-store.

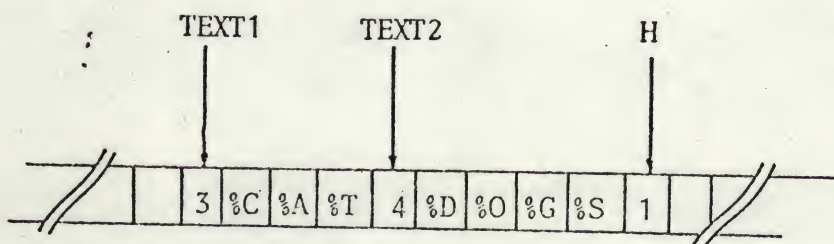
Vector assignation to the first identifier in the list has the effect that the value on the address of this identifier becomes the vector length. The values of the following identifiers are the consecutive element values of the assigned vector.

For example: after the execution of VECTOR LNTH,EL1,EL2,EL3,EL4; ...; LNTH *= VEC(A EL1,A EL2,%H,%L); the value of LNTH is 4, of EL1 and EL2 the values are equal to the addresses of themselves, and EL3 has the character value H and EL4 the character value L.

The ARRAY operator reserves store locations in the X-store for vector assignation into this store as the declarational operator VECTOR does, but after the array identifier a number is given which denotes the number of X-store locations to reserve following the location pointed by the identifier. No identifier name refers to these 'element locations'.

For example: the elaboration of the program part

ARRAY TEXT1(3),TEXT2(4) ; H:=1 ; TEXT1 *="CAT" ; TEXT2 *="DOGS" ;
gives the following X-store lay-out :



January 1915

Received of Mr. J. H. [unclear] the sum of [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

for [unclear] [unclear] [unclear]

the sum of [unclear] [unclear] [unclear]

The FORMAL operator allocates the identifiers following the operator in the formal identifier list in the Y-store.

These identifiers are used as named formal parameters of subroutines. These identifiers refer to the locations which are filled in with the actual parameter values at the call of the subroutine.

Note : The declaration of the formal parameters must be done before a declaration of local variables.

For example: the McCarthy's F91 function (4)

```
SUBR(F91 : FORMAL NUMBER;
      ( NUMBER>100 THEN NUMBER-10 ELSE F91(F91(NUMBER+11))));
```

This function generates as result for all numbers less than 102 the number 91 recursively.

The LOCAL declarational operator allocates following identifiers to the Y-store. These identifiers can be seen as variables for local usage. For example the interchange subroutine:

```
SUBR( CHANGE: FORMAL X,Y ; LOCAL HELP ;
      HELP:= C X ; C X := C Y ; C Y := HELP );
```

This can be written shorter as :

```
SUBR(CHANGE : FORMAL X,Y ; C @[Y,X]:=@C@[X,Y] );
```

The declarational operators LABEL and GLOBAL are treated at page 46 .

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950
JAMES H. HARRIS
JAMES H. HARRIS

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950
JAMES H. HARRIS
JAMES H. HARRIS

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950
JAMES H. HARRIS
JAMES H. HARRIS

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950
JAMES H. HARRIS
JAMES H. HARRIS

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950
JAMES H. HARRIS
JAMES H. HARRIS

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950
JAMES H. HARRIS
JAMES H. HARRIS

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950
JAMES H. HARRIS
JAMES H. HARRIS

input and output.

All transput operations take place on files.

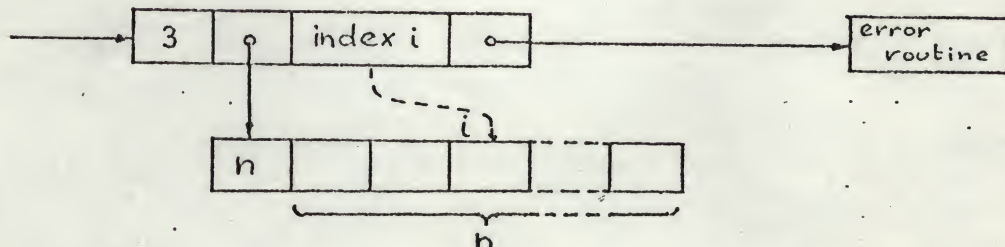
A file consists of a vector having three (or more) elements.

The first element is (the address of) a vector which contains (on input) or will contain (on output) the values transferred .

The second element is an integer file index which acts as the subscript for the file or buffer vector given in the first vector element.

The third element is (the address of) an escape subroutine.

This subroutine is called if the source or sink file is exhausted, or if $i=0$.



If the error subroutine is called the first formal parameter will be the file address. If the error occurs at an output operation, the second transferred parameter of the error routine will be the value to output.

The value returned by an input operation is the value of the transput operation.

If a transput operation is done without error , the file index is incremented with one.

The user is free to initiate other transput operations or alter the file or file index in any way he wishes by means of the escape or error subroutine.

By means of this file structure it is possible to impliment buffer structures for input and output operations.

THEORY

The first part of the theory is the definition of the function $f(x)$ and the function $F(x)$. The function $f(x)$ is defined as the function which is continuous at x and has a unique tangent at x . The function $F(x)$ is defined as the function which is continuous at x and has a unique tangent at x .



The second part of the theory is the definition of the function $f(x)$ and the function $F(x)$. The function $f(x)$ is defined as the function which is continuous at x and has a unique tangent at x . The function $F(x)$ is defined as the function which is continuous at x and has a unique tangent at x .

The third part of the theory is the definition of the function $f(x)$ and the function $F(x)$. The function $f(x)$ is defined as the function which is continuous at x and has a unique tangent at x . The function $F(x)$ is defined as the function which is continuous at x and has a unique tangent at x .

The fourth part of the theory is the definition of the function $f(x)$ and the function $F(x)$. The function $f(x)$ is defined as the function which is continuous at x and has a unique tangent at x . The function $F(x)$ is defined as the function which is continuous at x and has a unique tangent at x .

The fifth part of the theory is the definition of the function $f(x)$ and the function $F(x)$. The function $f(x)$ is defined as the function which is continuous at x and has a unique tangent at x . The function $F(x)$ is defined as the function which is continuous at x and has a unique tangent at x .

The monadic operator R obtains a value from a file (vector address).

For example: the elaboration of :

```
( FILE:= VEC("INPUT",2, SUBR(FAIL: .....)) ; R FILE )
```

delivers the character value %N, and the file will be

changed to VEC("INPUT",3,SUBR(FAIL:))

The monadic operator AP converts its operand, a file (vector address), into a conventional address. Assignment to this address with the \leftarrow assignment operator places the assigned value into the file.

A vector assignment to this file causes the transfer of all the vector data elements. (the length is not transferred)

For example: after the three next operations

```
FILE:= VEC(20+0 , 1 , SUBR(FAIL: ....) );
```

```
P FILE *= "INPUT AND " ; P FILE *= "OUTPUT" ;
```

the result file will be :

```
VEC("INPUT AND OUTPUT",17, SUBR(FAIL: .... )) ; ,
```

where "input and output" denotes the vector of twenty elements filled with the 16 transferred characters.

As it can be seen in the example above, the combination of the two operators C and AP may be shortened to P .

input and output to external devices.

By convention the files with the vector addresses 1 to 4 are associated with external devices. They are seen as vectors of length zero with an infinite capacity.

The files addresses 1 and 2 are reserved for the teletype input and output. (not buffered)

The first of these is the fact that the
the second is the fact that the
the third is the fact that the

the fourth is the fact that the
the fifth is the fact that the
the sixth is the fact that the

the seventh is the fact that the
the eighth is the fact that the
the ninth is the fact that the

the tenth is the fact that the
the eleventh is the fact that the
the twelfth is the fact that the

the thirteenth is the fact that the
the fourteenth is the fact that the
the fifteenth is the fact that the

For example : the execution of `P1:=%L ;` causes the printing of a character L on the teleprinter, and the elaboration of `X:=R 1 ;` is the reading of character of the teletype keyboard and assigning that character value to X .

The file addresses 3 and 4 are used for the input and output by means of buffered device handlers. These input and output devices are given to the system in the file command string. (see page 96)

The input and output file channels 1 and 3 are specifically for character transput : on input, a line-feed character is deleted from the input string, but the echo of channel 1 on the teletype printer generates this line-feed character after every carriage-return character.

On output a line-feed character is generated after every carriage-return character, as well as on channel 1 as on channel 3.

The input and output file channels 2 and 4 are specifically for other transput operations as for example transput operations of binary code .

Transput operations on these channels do not perform any character generation, dropping or echoing .

other file operators

The DECIN operator will read from its file operand a possibly signed decimal integer of at most four digits. The integer is terminated by any character which is not a digit.

The returned value after the execution of this operator on a file is the integer converted into a number of twelve bits length.

The OCTIN operator will read from its file operand an octal integer of at most 4 octal digits. Termination is also done by

a character which is not an (octal) digit.

The DIGIT operator will read its file operand consecutively until a digit is detected. The value returned by this operand is this digit character.

Note: to obtain a numerical digit value from the digit character subtract the character value %0.

The ALPHA operator acts as the DIGIT operator, except that the detection is done on the first letter or digit character, and this character is returned.

The CHAR operator reads from its file operand a character. This operator skips all control symbols, the rubout symbol, the carriage-return character and the line-feed character, spaces and the altmode character.

Examples : P 3:= "READ NUMBER"; will cause the string
READ NUMBER to be transferred to the output buffer
of the output device given in the file command string.

P 1:= ALPHA 3 ; will cause the first digit or letter
in the input buffer of the device given in the file command
string to be printed on the teleprinter.

P 1:= ALPHA VEC("\$%&((\$\$%3A&", 1, SUBR(FAIL: HALT))
will cause, if executed, the character 3 to be printed,
and file index is changed to 10 .

output conversion operators.

The DECS operator converts its operand, a number, into a string of 5 characters. The first one is a space for positive numbers, or a minus sign (-) for negative numbers respectively. The following four characters are digits. This string is the signed decimal representation

THE HISTORY OF THE
CITY OF BOSTON

FROM THE FIRST SETTLEMENT
TO THE PRESENT TIME

BY
JOHN B. BOWEN

IN TWO VOLUMES.
VOL. I.

BOSTON:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

NEW-YORK:

WILLIAM B. BOWEN, 1850.

of the operand at the right hand side of the operator.

The OCTS operator produces a string of four octal digits of its operand. This string is the octal representation of the operand.

For example: the elaboration of the program part

```
SUM:= DECIN (1) + DECIN (1) ; P 1:=SP ;  
P 1*= DECS(SUM)." ".OCTS(SUM).VEC(LN) ;
```

will read two decimal integer (possible signed) of the teletype keyboard, and prints on the teletype the sum in decimal and in octal notations.

procedures.

There are three proceduring operators : PROG, SUBR and EXPR .

All proceduring operators procedure their secondaries, which means that at the elaboration of the PROG, SUBR and EXPR operator the secondaries are not evaluated but are skipped. These secondaries can be executed by calling them with special call operators.

The value of a proceduring operator working on a secondary, which is the value needed for the call operator, is the address of the first instruction of the secondary in the memory.

The proceduring operator EXPR procedures its secondary, which denotes an expression which must be evaluated at the moment of call.

The Y-store referred to within an EXPR is the same is procedured without a change in environment.

The proceduring operator SUBR is used as procedure in the ALGOL way.

Every call of a subroutine (or SUBR secondary) creates a new environment of Y-store locations which are used for parameter transfer.

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
DEPARTMENT OF CHEMISTRY

REPORT OF THE
COMMISSIONER OF THE
BUREAU OF CHEMISTRY
FOR THE YEAR 1900
AND THE
PROGRESS OF THE
BUREAU DURING THE
YEAR 1901

CHICAGO
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
1902

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
DEPARTMENT OF CHEMISTRY
CHICAGO, ILL.

Within the secondary of a SUBR operator three operators are defined working on Y-stores, the M , N and Z operators. They are treated at page 71 at the subroutine call.

Note : the address returned by a EXPR or a SUBR is equal to the value of a label pointing on the first executional statement of the secondary, so these secondaries can be called by using this label.

The proceduring operator PROG procedures its secondary as a program. At the evaluation of it, an own X-store and an own Y-store is created. No references and data transfer is possible with the outer scope (environment) except by means of the W-store locations.

calling of expressions.

Expressions, which are the secondaries of EXPR operators, are called by means of the V operator. This operator calls the secondary which starts at the address given by the operand. No change of environment of Y-store locations is made on such a V call. For example : execution of the following program part:

```
( EXPR(IN: P 1*="NUMBERS: "; DECIN(1)+DECIN(1) );
```

```
  WHILE TRUE DO
```

```
    P1 := LN ; P1*= DECS( V IN ) ; P 1:=LN
```

```
  OD )
```

causes the sum of the read numbers printed as a decimal number to be printed on the teletype.



The address delivered by the EXPR operator is voided within this example, but can be used as follows :

```
( X:= EXPR( P 1 *="NUMBERS:" ; DECIN (1) + DECIN (1) ) ;
  WHILE TRUE DO
    P 1:= LN ; P 1:= DECS( V (X) ) ; P 1:= LN
  OD ) ;
```

Note : the address value stored in X is called by V, but because of the fact that V expects a label behind it, X must be enclosed between parenthesis. (see implicit definition, page 44)

calling of subroutines.

The secondary of the proceduring SUBR can be called with the \$ sum operator. This operator works on a vecotr with address of the secondary of the SUBR as first element.

The elements following the address element are the actual parameters. The value returned after calling a subroutine is the value delivered by its secondary. If this secondary is void, the returned value after calling the subroutine is undefined.

A subroutine can be also called as a function. This can be done if the SUBR operates on a labelled secondary.

The function call is as follows:

label identifier(actual parameter list).

For example: the following calls are equal :

```
: SUBR(CALL: FORMAL PAR ; IF PAR=0 THEN 1 ELSE
  PAR * CALL(PAR-1) FI ) ;
X:=CALL(5) ; Y:=$ VEC(CALL,5) ; Z:=..CALL..(5) ; .
```

The ..label identifier.. operator is a user defined operator. .

THE UNIVERSITY OF CHICAGO
LIBRARY

1000 S. EAST ASIAN LIBRARY
CHICAGO, ILL. 60607

DATE 10/10/1968

FROM THE UNIVERSITY OF CHICAGO
LIBRARY

TO THE UNIVERSITY OF CHICAGO
LIBRARY

FROM THE UNIVERSITY OF CHICAGO
LIBRARY

TO THE UNIVERSITY OF CHICAGO
LIBRARY

FROM THE UNIVERSITY OF CHICAGO
LIBRARY

TO THE UNIVERSITY OF CHICAGO
LIBRARY

FROM THE UNIVERSITY OF CHICAGO
LIBRARY

TO THE UNIVERSITY OF CHICAGO
LIBRARY

FROM THE UNIVERSITY OF CHICAGO
LIBRARY

TO THE UNIVERSITY OF CHICAGO
LIBRARY

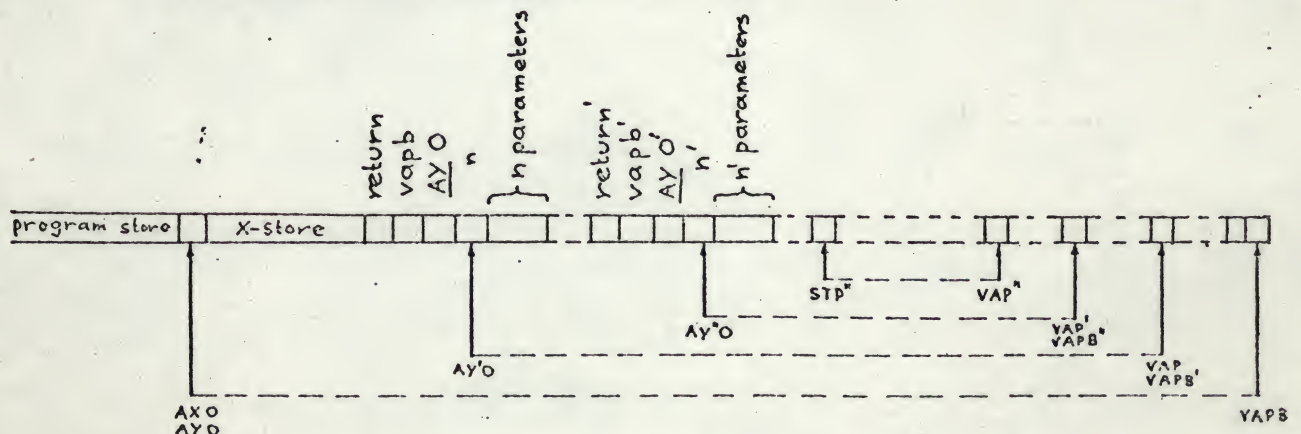
This operator can be used either monadic or dyadic, depending on the number of formal parameters used in the subroutine.

in the program day of the date (page 132) this way of calling is used for a modulo function.

At calling a subroutine the following actions are taken by the system :

1. the return address is pushed on the stack top.
2. the vector bottom pointer (VAPB) is pushed on the stack.
3. the address of Y 0 (= AY 0) is pushed on the stack.
4. the vector bottom pointer is moved towards the vector stack top pointer (VAP) or : $VAPB := VAP$.
5. the AY 0 pointer is moved to the location following the last stacked value (AY 0 of the last environment)
6. the actual parameters are transferred as a vector into a newly created Y-store.
7. The stackpointer (STP) is moved to the location address following the location of the last transferred parameters .

So every subroutine call creates a new environment. In the next lay-out a diagram is made of the memory after a call of a nested subroutine. Every ' denotes a call of the subroutine, and the pointers with two single quotes denotes the pointers which belongs to the environment of the second call.



THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

TO THE HONORABLE CHAIRMAN OF THE BOARD OF TRUSTEES
OF THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

As it can be seen from the store lay-out Y -1 denotes the old AY 0 pointer and Y -2 the old VAPB pointer.

The monadic operator M applied on an Y-store variable or an Y operator exposes the previous proceduring secondary environment.

M acts as a parameterless procedure which is called.

This means that M Y 2 refers to the second Y-store of the environment which called the current environment.

The combination of the M operator applied to the Y operator is provided by the N operator.

Thus the following subroutine can be defined :

```
SUBR(FAC: IF N1=0 THEN 1 ELSE N1*FAC( EXPR(N1-1))FI ) ;
```

this corresponds to the ALGOL 60 program :

```
integer procedure fac(i) ; begin fac:=if i=0 then 1 else i*fac(i-1)
                               end ;
```

As it can be seen, in the example, the value given in the recursive call is procedured with EXPR to enclose the value in an environment.

The Z operator is the combination of the C operator applied to an Y operator. (Z \equiv CY)

This operator is specifically used if transferred parameters of the subroutine are of address mode.

At the end of the elaboration of a subroutine call, all pointers stacked at the call are restored. The vector area top pointer (VAP) keeps its value.

The loader is a subroutine too. So it can be called from the program (see page 102).

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES

REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO
FOR THE YEAR 1900
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
CHICAGO, ILL., 1901

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO
FOR THE YEAR 1900
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
CHICAGO, ILL., 1901

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO
FOR THE YEAR 1900
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
CHICAGO, ILL., 1901

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO
FOR THE YEAR 1900
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
CHICAGO, ILL., 1901

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO
FOR THE YEAR 1900
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
CHICAGO, ILL., 1901

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO
FOR THE YEAR 1900
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
CHICAGO, ILL., 1901

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO
FOR THE YEAR 1900
PUBLISHED BY THE
UNIVERSITY OF CHICAGO PRESS
CHICAGO, ILL., 1901

calling of programs.

The operator CALLPR is used for calling a program. This operator takes the address of the first instruction of the called program, delivered by the proceduring operator PROG , as its operand.

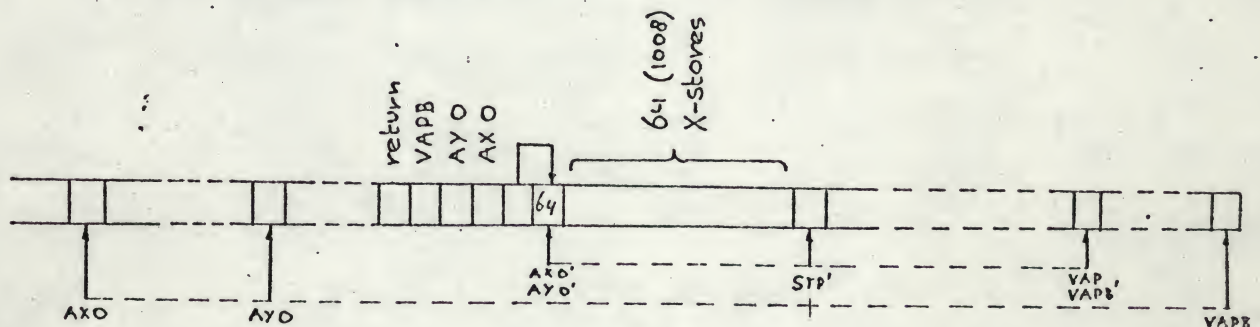
When a program is loaded by the loader, it can be called by operating the CALLPR operator on the address following the program buffer start address.

At calling a program, the following actions are taken :

1. the return address is pushed on the stack top.
2. the vector area bottom pointer is pushed on the stack.
3. the AY 0 pointer is pushed on the stack.
4. the vector area bottom pointer (VAPB) is moved to the vector area top pointer. (VAPB:=VAP)
5. the AX 0 pointer is pushed on the stack.
6. the value of the stackpointer (STP) added with one is pushed on the stack.
7. the AX 0 pointer is set to the location following the stack top.
the AY 0 pointer is set to this location too.
8. X 0 gets the value B 0100 and the stackpointer is incremented with this value. (declaration of 64 X-stores which belongs to this program)

The store lay-out after a program call is as follows :

(a ' denotes the pointer after the call)



THE UNIVERSITY OF CHICAGO

The first of the two main parts of the book is devoted to a study of the history of the theory of the origin of life. The second part is devoted to a study of the theory of the origin of the universe.

The first part of the book is devoted to a study of the history of the theory of the origin of life. The second part is devoted to a study of the theory of the origin of the universe.

The first part of the book is devoted to a study of the history of the theory of the origin of life. The second part is devoted to a study of the theory of the origin of the universe.

The first part of the book is devoted to a study of the history of the theory of the origin of life. The second part is devoted to a study of the theory of the origin of the universe.

1850
1851
1852

THE UNIVERSITY OF CHICAGO
CHICAGO, ILL.

As it can be seen from the store lay-out X-2 denotes the AX 0 address of the environment before the call, and X-3 the AY 0 address.

The program calls may be nested to any depth. Each successive call handing on to the called program the store resources it has left unused.

A program is left when a jump is made to the end of program routine. The resources used by the called program then are deallocated, and the program which called the inner program is reactivated.

The RESET operation is done for the vector stack used in the program by the program end routine ! .

The value returned by a CALLPR operation is undefined.

Data transfer between a program and an embedded called program is only possible by means of the W-store or common variable room. From inside of a called program there is also a possibility for data transfer to or from the X-store of the calling program by means of the AX 0 address of it, which is stored in the X-2 location as seen above..

goto.

The GOTO operator causes the next quaternary to the elaborated to be the one whose first instruction is located at the machine address which is given by the operand of the GOTO.

If the operand does not have the same value as could be possessed by some label, then the subsequent action is undefined.

Usually the operand will be a label, though it need not to be :

LVAL:= LABEL1 ; ; GOTO (LVAL) ; .

the first of these is the fact that the
the second is the fact that the

the third is the fact that the

the fourth is the fact that the

the fifth is the fact that the

the sixth is the fact that the

the seventh is the fact that the

the eighth is the fact that the

the ninth is the fact that the

the tenth is the fact that the

the eleventh is the fact that the

has the effect GOTO LABEL1 ; .

Notice that if an identifier is used after a GOTO and that identifier is not a label identifier, that this identifier has to be enclosed by parenthesis. (see identifiers, page 44)

miscellaneous.

The SKIP operation is a no-operation instruction. (B 7000 on the PDP-8)
This dummy instruction can be used for creating free locations in the programstore or can be used as a dummy instruction in a case conditional.

For example : CASE SWITCH IN SKIP,X+:1,Y+:2,SKIP OUT RESET ESAC ; .

The TABLE operator procedures its following operand.

The table items are stored in consecutive memory locations.

A string is stored as a vector of characters in this table store, which is embedded in the program store.

The item list of the CODE operator is seen as a list of directly coded, non-relocatable , machine instructions. These instructions are incorporated in the binary code as directly outputted data frames.

The SREG value gives the contents of the switch register of the computer.

For example : WHILE SREG/=1 DO wait-loop OD ; .

The exit or HALT operation (B 4000) will cause a jump to the error routine. This error routine will print an error line ("SYS ERR AT nnnn") and closes the output file before returning to the Monitor.



6.4 THE PREPROCESSOR.

The preprocessor is an one pass translator from Midial to Minial. Therefore the declarational operators are introduced in Midial to translate rigth identifiers on place where there is no clearness of how the identifier must be translated correct.

On the next pages the design of the identifier list of the preprocessor is given, followed by the flowcharts of the preprocessor.

After these pages the listing of the preprocessor written in Minial is added, but , because of its greater readability, the preprocessor version written in Midial (page 106) can be used for understanding the preprocessor itselfs.

The preprocessor minimum system, which is used by all the preprocessor versions, has only the following operator subroutines :

the + , - , / , * , HALT , RESET , CALLPR , V , \$, X , Y , AX , AY , P , R , C , COPY , DECS , GOTO , DCL , all assignation operators , HD , all relational operators , & , ! , < , all defined operator and becomes operators , EXPR , SUBR , PROG , and all CASE , WHILE and IF constructions.

The first of these is the fact that the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

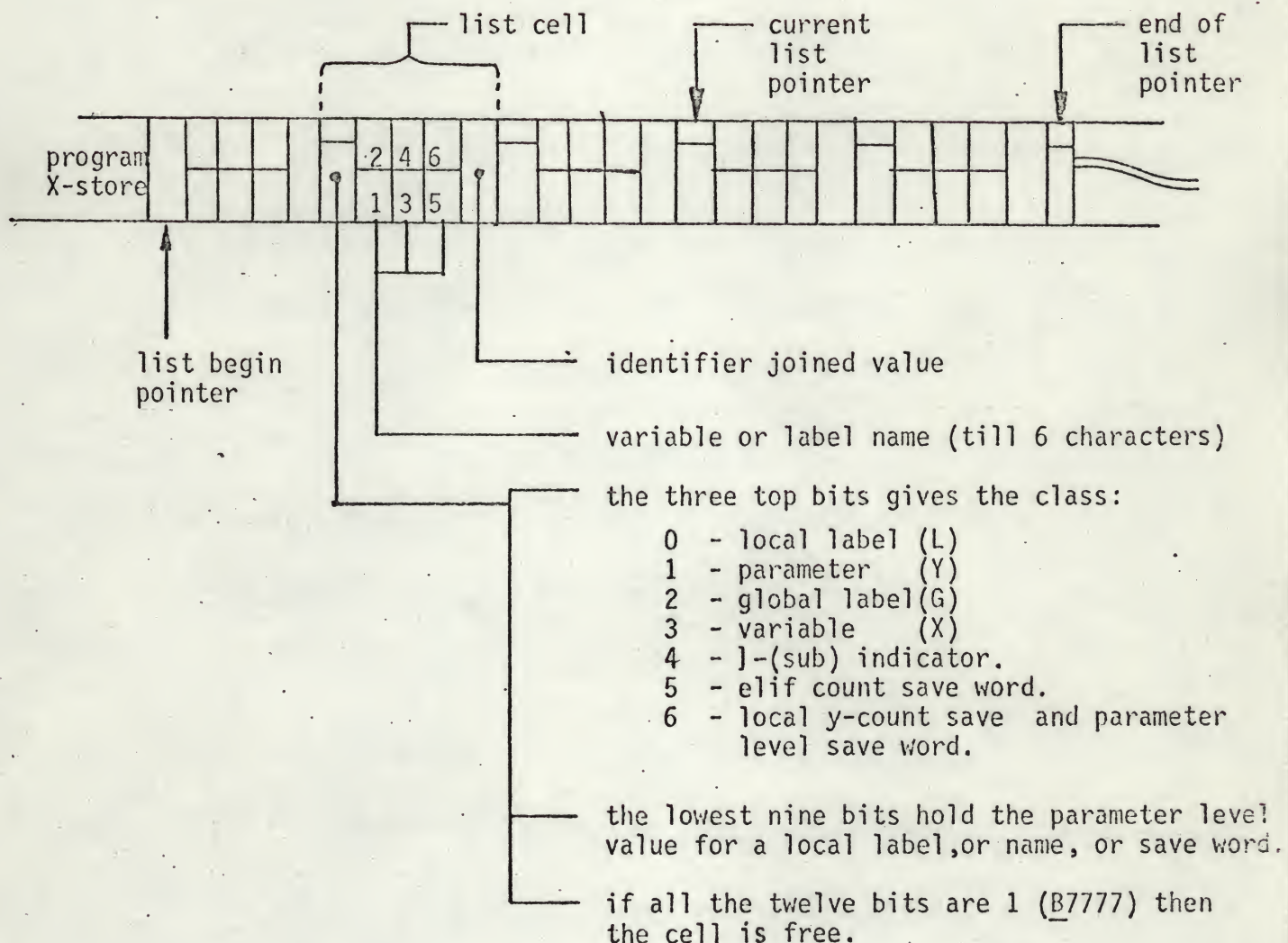
presented in the first part of the

presented in the first part of the

presented in the first part of the

presented in the first part of the

THE PREPROCESSOR IDENTIFIER LIST DESIGN.



Characters : six characters are packed in three memory locations for remembering an identifier name.

This is done by subtracting B0257 from the character value.

The digits 0 till 9 are so represented as B01 till B12 , and the alphabetical symbols A till Z are represented as B22 till B53 .

A zero represents a blank symbol. These zeroes fill not used character places .

1. The first part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom.

2. In the second part, we shall consider the question of the influence of the external magnetic field on the structure of the atom.

3. The third part of the paper is devoted to a discussion of the question of the influence of the external electric field on the structure of the atom.

4. In the fourth part, we shall consider the question of the influence of the external magnetic field on the structure of the atom.

5. The fifth part of the paper is devoted to a discussion of the question of the influence of the external electric field on the structure of the atom.

6. In the sixth part, we shall consider the question of the influence of the external magnetic field on the structure of the atom.

7. The seventh part of the paper is devoted to a discussion of the question of the influence of the external electric field on the structure of the atom.

8. In the eighth part, we shall consider the question of the influence of the external magnetic field on the structure of the atom.

9. The ninth part of the paper is devoted to a discussion of the question of the influence of the external electric field on the structure of the atom.

10. In the tenth part, we shall consider the question of the influence of the external magnetic field on the structure of the atom.

11. The eleventh part of the paper is devoted to a discussion of the question of the influence of the external electric field on the structure of the atom.

12. In the twelfth part, we shall consider the question of the influence of the external magnetic field on the structure of the atom.

Identifier joined value : is the value given to the identifier by the preprocessor when the identifier is declared implicitly or explicitly .

Sub-indicator : is, a list cell which indicates that, if a close token is met which closes the level where the sub-indicator is created, this close token must be converted to a sub (]) character. .
This occurs at the closing of a function call or at the closing of a vector given by VEC(.

Elif count save word : This save word holds in the location where normally the first two characters are packed, the count of the number of ELIF's on the outer level (or scope) .

Local Y-count save
and parameter level save word :

This word contains the Y-count (character 1 and 2 and the parameter level (on character 3 and 4) of the outer parameter level. This word is generated at the entrance of a SUBR or of an EXPR .

When there are no more free list cells within the identifier list at the time of the declaration of an identifier or of a level word, the end list pointer is moved 5 positions (or memory locations) further and the identifier or level word has been placed in this new created cell.

This can be done until the stack begin is reached.

THE FIRST PART OF THE

THE SECOND PART OF THE

THE THIRD PART OF THE

THE FOURTH PART OF THE

THE FIFTH PART OF THE

THE SIXTH PART OF THE

THE SEVENTH PART OF THE

THE EIGHTH PART OF THE

THE NINTH PART OF THE

THE TENTH PART OF THE

THE ELEVENTH PART OF THE

THE TWELFTH PART OF THE

THE THIRTEENTH PART OF THE

THE FOURTEENTH PART OF THE

THE FIFTEENTH PART OF THE

THE PREPROCESSOR

DRIVER

lcnt := xcnt := gcnt := 20 ; scope := elifcnt := 0 ;
parlev := 1 ; flgfound := false

initialise Listpointers ; create 1 empty list cell

declare the maximal X-store for the list

readchar (-acter)

output "(DCL(100);"

repeat while inputchar \neq CNTRL Z -file end-

INPUT CHARACTER									
symbol	alpha symbol	# symbol	SP, LN, tab, lf, null, form feed -symbol	(symbol) symbol	; symbol	%-symbol	" symbol	an other symbol
midialsym	name	comment	sp, lntab	push-scope	pop-scope	flgcomma := true	resetflg	until next	resetflg
							next-char	next-char	
next char									

comma skip

scope > 0 ?

+ close tokens missing

output warning

while scope > 0

output a)-symbol

pop scope (scope := 1)

output a)-symbol and a ;-symbol followed by the file end symbol.

1. Introduction
The purpose of this study is to investigate the effects of various factors on the growth of plants.

2. <u>Methodology</u>	
The experiment was conducted in a controlled environment over a period of 12 weeks.	
The following factors were investigated:	
a. Light intensity	
b. Water availability	
c. Soil pH	
d. Nutrient concentration	
e. Temperature	
f. Humidity	
g. Air circulation	
h. CO2 concentration	
i. Root system development	
j. Leaf area and chlorophyll content	
k. Stomatal conductance	
l. Transpiration rate	
m. Biomass accumulation	
n. Root-to-shoot ratio	
o. Leaf-to-root ratio	
p. Root length	
q. Root diameter	
r. Root branching	
s. Root hair density	
t. Root hair length	
u. Root hair diameter	
v. Root hair branching	
w. Root hair density	
x. Root hair length	
y. Root hair diameter	
z. Root hair branching	
aa. Root hair density	
ab. Root hair length	
ac. Root hair diameter	
ad. Root hair branching	
ae. Root hair density	
af. Root hair length	
ag. Root hair diameter	
ah. Root hair branching	
ai. Root hair density	
aj. Root hair length	
ak. Root hair diameter	
al. Root hair branching	
am. Root hair density	
an. Root hair length	
ao. Root hair diameter	
ap. Root hair branching	
aq. Root hair density	
ar. Root hair length	
as. Root hair diameter	
at. Root hair branching	
au. Root hair density	
av. Root hair length	
aw. Root hair diameter	
ax. Root hair branching	
ay. Root hair density	
az. Root hair length	
ba. Root hair diameter	
bb. Root hair branching	
bc. Root hair density	
bd. Root hair length	
be. Root hair diameter	
bf. Root hair branching	
bg. Root hair density	
bh. Root hair length	
bi. Root hair diameter	
bj. Root hair branching	
bk. Root hair density	
bl. Root hair length	
bm. Root hair diameter	
bn. Root hair branching	
bo. Root hair density	
bp. Root hair length	
bq. Root hair diameter	
br. Root hair branching	
bs. Root hair density	
bt. Root hair length	
bu. Root hair diameter	
bv. Root hair branching	
bw. Root hair density	
bx. Root hair length	
by. Root hair diameter	
bz. Root hair branching	
ca. Root hair density	
cb. Root hair length	
cc. Root hair diameter	
cd. Root hair branching	
ce. Root hair density	
cf. Root hair length	
cg. Root hair diameter	
ch. Root hair branching	
ci. Root hair density	
cj. Root hair length	
ck. Root hair diameter	
cl. Root hair branching	
cm. Root hair density	
cn. Root hair length	
co. Root hair diameter	
cp. Root hair branching	
cq. Root hair density	
cr. Root hair length	
cs. Root hair diameter	
ct. Root hair branching	
cu. Root hair density	
cv. Root hair length	
cw. Root hair diameter	
cx. Root hair branching	
cy. Root hair density	
cz. Root hair length	
da. Root hair diameter	
db. Root hair branching	
dc. Root hair density	
dd. Root hair length	
de. Root hair diameter	
df. Root hair branching	
dg. Root hair density	
dh. Root hair length	
di. Root hair diameter	
dj. Root hair branching	
dk. Root hair density	
dl. Root hair length	
dm. Root hair diameter	
dn. Root hair branching	
do. Root hair density	
dp. Root hair length	
dq. Root hair diameter	
dr. Root hair branching	
ds. Root hair density	
dt. Root hair length	
du. Root hair diameter	
dv. Root hair branching	
dw. Root hair density	
dx. Root hair length	
dy. Root hair diameter	
dz. Root hair branching	
ea. Root hair density	
eb. Root hair length	
ec. Root hair diameter	
ed. Root hair branching	
ee. Root hair density	
ef. Root hair length	
eg. Root hair diameter	
eh. Root hair branching	
ei. Root hair density	
ej. Root hair length	
ek. Root hair diameter	
el. Root hair branching	
em. Root hair density	
en. Root hair length	
eo. Root hair diameter	
ep. Root hair branching	
eq. Root hair density	
er. Root hair length	
es. Root hair diameter	
et. Root hair branching	
eu. Root hair density	
ev. Root hair length	
ew. Root hair diameter	
ex. Root hair branching	
ey. Root hair density	
ez. Root hair length	
fa. Root hair diameter	
fb. Root hair branching	
fc. Root hair density	
fd. Root hair length	
fe. Root hair diameter	
ff. Root hair branching	
fg. Root hair density	
fh. Root hair length	
fi. Root hair diameter	
fj. Root hair branching	
fk. Root hair density	
fl. Root hair length	
fm. Root hair diameter	
fn. Root hair branching	
fo. Root hair density	
fp. Root hair length	
fq. Root hair diameter	
fr. Root hair branching	
fs. Root hair density	
ft. Root hair length	
fu. Root hair diameter	
fv. Root hair branching	
fw. Root hair density	
fx. Root hair length	
fy. Root hair diameter	
fz. Root hair branching	
ga. Root hair density	
gb. Root hair length	
gc. Root hair diameter	
gd. Root hair branching	
ge. Root hair density	
gf. Root hair length	
gg. Root hair diameter	
gh. Root hair branching	
gi. Root hair density	
gj. Root hair length	
gk. Root hair diameter	
gl. Root hair branching	
gm. Root hair density	
gn. Root hair length	
go. Root hair diameter	
gp. Root hair branching	
gq. Root hair density	
gr. Root hair length	
gs. Root hair diameter	
gt. Root hair branching	
gu. Root hair density	
gv. Root hair length	
gw. Root hair diameter	
gx. Root hair branching	
gy. Root hair density	
gz. Root hair length	
ha. Root hair diameter	
hb. Root hair branching	
hc. Root hair density	
hd. Root hair length	
he. Root hair diameter	
hf. Root hair branching	
hg. Root hair density	
hh. Root hair length	
hi. Root hair diameter	
hj. Root hair branching	
hk. Root hair density	
hl. Root hair length	
hm. Root hair diameter	
hn. Root hair branching	
ho. Root hair density	
hp. Root hair length	
hq. Root hair diameter	
hr. Root hair branching	
hs. Root hair density	
ht. Root hair length	
hu. Root hair diameter	
hv. Root hair branching	
hw. Root hair density	
hx. Root hair length	
hy. Root hair diameter	
hz. Root hair branching	
ia. Root hair density	
ib. Root hair length	
ic. Root hair diameter	
id. Root hair branching	
ie. Root hair density	
if. Root hair length	
ig. Root hair diameter	
ih. Root hair branching	
ii. Root hair density	
ij. Root hair length	
ik. Root hair diameter	
il. Root hair branching	
im. Root hair density	
in. Root hair length	
io. Root hair diameter	
ip. Root hair branching	
iq. Root hair density	
ir. Root hair length	
is. Root hair diameter	
it. Root hair branching	
iu. Root hair density	
iv. Root hair length	
iw. Root hair diameter	
ix. Root hair branching	
iy. Root hair density	
iz. Root hair length	
ja. Root hair diameter	
jb. Root hair branching	
jc. Root hair density	
jd. Root hair length	
je. Root hair diameter	
jf. Root hair branching	
jg. Root hair density	
jh. Root hair length	
ji. Root hair diameter	
jj. Root hair branching	
jk. Root hair density	
jl. Root hair length	
jm. Root hair diameter	
jn. Root hair branching	
jo. Root hair density	
jp. Root hair length	
jq. Root hair diameter	
jr. Root hair branching	
js. Root hair density	
jt. Root hair length	
ju. Root hair diameter	
jv. Root hair branching	
jw. Root hair density	
jx. Root hair length	
jy. Root hair diameter	
jz. Root hair branching	
ka. Root hair density	
kb. Root hair length	
kc. Root hair diameter	
kd. Root hair branching	
ke. Root hair density	
kf. Root hair length	
kg. Root hair diameter	
kh. Root hair branching	
ki. Root hair density	
kj. Root hair length	
kk. Root hair diameter	
kl. Root hair branching	
km. Root hair density	
kn. Root hair length	
ko. Root hair diameter	
kp. Root hair branching	
kq. Root hair density	
kr. Root hair length	
ks. Root hair diameter	
kt. Root hair branching	
ku. Root hair density	
kv. Root hair length	
kw. Root hair diameter	
kx. Root hair branching	
ky. Root hair density	
kz. Root hair length	
la. Root hair diameter	
lb. Root hair branching	
lc. Root hair density	
ld. Root hair length	
le. Root hair diameter	
lf. Root hair branching	
lg. Root hair density	
lh. Root hair length	
li. Root hair diameter	
lj. Root hair branching	
lk. Root hair density	
ll. Root hair length	
lm. Root hair diameter	
ln. Root hair branching	
lo. Root hair density	
lp. Root hair length	
lq. Root hair diameter	
lr. Root hair branching	
ls. Root hair density	
lt. Root hair length	
lu. Root hair diameter	
lv. Root hair branching	
lw. Root hair density	
lx. Root hair length	
ly. Root hair diameter	
lz. Root hair branching	
ma. Root hair density	
mb. Root hair length	
mc. Root hair diameter	
md. Root hair branching	
me. Root hair density	
mf. Root hair length	
mg. Root hair diameter	
mh. Root hair branching	
mi. Root hair density	
mj. Root hair length	
mk. Root hair diameter	
ml. Root hair branching	
mo. Root hair density	
mp. Root hair length	
mq. Root hair diameter	
mr. Root hair branching	
ms. Root hair density	
mt. Root hair length	
mu. Root hair diameter	
mv. Root hair branching	
mw. Root hair density	
mx. Root hair length	
my. Root hair diameter	
mz. Root hair branching	
na. Root hair density	
nb. Root hair length	
nc. Root hair diameter	
nd. Root hair branching	
ne. Root hair density	
nf. Root hair length	
ng. Root hair diameter	
nh. Root hair branching	
ni. Root hair density	
nj. Root hair length	
nk. Root hair diameter	
nl. Root hair branching	
no. Root hair density	
np. Root hair length	
nq. Root hair diameter	
nr. Root hair branching	
ns. Root hair density	
nt. Root hair length	
nu. Root hair diameter	
nv. Root hair branching	
nw. Root hair density	
nx. Root hair length	
ny. Root hair diameter	
nz. Root hair branching	
oa. Root hair density	
ob. Root hair length	
oc. Root hair diameter	
od. Root hair branching	
oe. Root hair density	
of. Root hair length	
og. Root hair diameter	
oh. Root hair branching	
oi. Root hair density	
oj. Root hair length	
ok. Root hair diameter	
ol. Root hair branching	
om. Root hair density	
on. Root hair length	
oo. Root hair diameter	
op. Root hair branching	
oq. Root hair density	
or. Root hair length	
os. Root hair diameter	
ot. Root hair branching	
ou. Root hair density	
ov. Root hair length	
ow. Root hair diameter	
ox. Root hair branching	
oy. Root hair density	
oz. Root hair length	
pa. Root hair diameter	
pb. Root hair branching	
pc. Root hair density	
pd. Root hair length	
pe. Root hair diameter	
pf. Root hair branching	
pg. Root hair density	
ph. Root hair length	
pi. Root hair diameter	
pj. Root hair branching	
pk. Root hair density	
pl. Root hair length	
pm. Root hair diameter	
pn. Root hair branching	
po. Root hair density	
pp. Root hair length	
pq. Root hair diameter	
pr. Root hair branching	
ps. Root hair density	
pt. Root hair length	
pu. Root hair diameter	
pv. Root hair branching	
pw. Root hair density	
px. Root hair length	
py. Root hair diameter	
pz. Root hair branching	
qa. Root hair density	
qb. Root hair length	
qc. Root hair diameter	
qd. Root hair branching	
qe. Root hair density	
qf. Root hair length	
qg. Root hair diameter	
qh. Root hair branching	
qi. Root hair density	
qj. Root hair length	
qk. Root hair diameter	
ql. Root hair branching	
qm. Root hair density	
qn. Root hair length	
qo. Root hair diameter	
qp. Root hair branching	
qq. Root hair density	
qr. Root hair length	
qs. Root hair diameter	
qt. Root hair branching	
qu. Root hair density	
qv. Root hair length	
qw. Root hair diameter	
qx. Root hair branching	
qy. Root hair density	
qz. Root hair length	
ra. Root hair diameter	
rb. Root hair branching	
rc. Root hair density	
rd. Root hair length	
re. Root hair diameter	
rf. Root hair branching	
rg. Root hair density	
rh. Root hair length	
ri. Root hair diameter	
rj. Root hair branching	
rk. Root hair density	
rl. Root hair length	
rm. Root hair diameter	
rn. Root hair branching	
ro. Root hair density	
rp. Root hair length	
rq. Root hair diameter	
rr. Root hair branching	
rs. Root hair density	
rt. Root hair length	
ru. Root hair diameter	
rv. Root hair branching	
rw. Root hair density	
rx. Root hair length	
ry. Root hair diameter	
rz. Root hair branching	
sa. Root hair density	
sb. Root hair length	
sc. Root hair diameter	
sd. Root hair branching	
se. Root hair density	
sf. Root hair length	
sg. Root hair diameter	
sh. Root hair branching	
si. Root hair density	
sj. Root hair length	
sk. Root hair diameter	
sl. Root hair branching	
sm. Root hair density	
sn. Root hair length	
so. Root hair diameter	
sp. Root hair branching	
sq. Root hair density	
sr. Root hair length	
ss. Root hair diameter	
st. Root hair branching	
su. Root hair density	
sv. Root hair length	
sw. Root hair diameter	
sx. Root hair branching	
sy. Root hair density	
sz. Root hair length	
ta. Root hair diameter	
tb. Root hair branching	
tc. Root hair density	
td. Root hair length	
te. Root hair diameter	
tf. Root hair branching	
tg. Root hair density	
th. Root hair length	
ti. Root hair diameter	
tj. Root hair branching	
tk. Root hair density	
tl. Root hair length	
tm. Root hair diameter	
tn. Root hair branching	
to. Root hair density	
tp. Root hair length	
tq. Root hair diameter	
tr. Root hair branching	
ts. Root hair density	
tt. Root hair length	
tu. Root hair diameter	
tv. Root hair branching	
tw. Root hair density	
tx. Root hair length	
ty. Root hair diameter	
tz. Root hair branching	
ua. Root hair density	
ub. Root hair length	
uc. Root hair diameter	
ud. Root hair branching	
ue. Root hair density	
uf. Root hair length	
ug. Root hair diameter	
uh. Root hair branching	
ui. Root hair density	
uj. Root hair length	
uk. Root hair diameter	
ul. Root hair branching	
um. Root hair density	
un. Root hair length	
uo. Root hair diameter	
up. Root hair branching	
uq. Root hair density	
ur. Root hair length	
us. Root hair diameter	
ut. Root hair branching	
uu. Root hair density	
uv. Root hair length	
uw. Root hair diameter	
ux. Root hair branching	
uy. Root hair density	
uz. Root hair length	
va. Root hair diameter	
vb. Root hair branching	
vc. Root hair density	
vd. Root hair length	
ve. Root hair diameter	
vf. Root hair branching	
vg. Root hair density	
vh. Root hair length	
vi. Root hair diameter	
vj. Root hair branching	
vk. Root hair density	
vl. Root hair length	
vm. Root hair diameter	
vn. Root hair branching	
vo. Root hair density	
vp. Root hair length	
vq. Root hair diameter	
vr. Root hair branching	
vs. Root hair density	
vt. Root hair length	
vu. Root hair diameter	
vv. Root hair branching	
vw. Root hair density	
vx. Root hair length	
vy. Root hair diameter	
vz. Root hair branching	
wa. Root hair density	
wb. Root hair length	
wc. Root hair diameter	
wd. Root hair branching	
we. Root hair density	
wf. Root hair length	
wg. Root hair diameter	
wh. Root hair branching	
wi. Root hair density	
wj. Root hair length	
wk. Root hair diameter	
wl. Root hair branching	
wm. Root hair density	
wn. Root hair length	
wo. Root hair diameter	
wp. Root hair branching	
wq. Root hair density	
wr. Root hair length	
ws. Root hair diameter	
wt. Root hair branching	
wu. Root hair density	
wv. Root hair length	
ww. Root hair diameter	
wx. Root hair branching	
wy. Root hair density	
wz. Root hair length	
xa. Root hair diameter	
xb. Root hair branching	
xc. Root hair density	
xd. Root hair length	
xe. Root hair diameter	
xf. Root hair branching	
xg. Root hair density	
xh. Root hair length	
xi. Root hair diameter	
xj. Root hair branching	
xk. Root hair density	
xl. Root hair length	
xm. Root hair diameter	
xn. Root hair branching	
xo. Root hair density	
xp. Root hair length	
xq. Root hair diameter	
xr. Root hair branching	
xs. Root hair density	
xt. Root hair length	
xu. Root hair diameter	
xv. Root hair branching	
xw. Root hair density	
xx. Root hair length	
xy. Root hair diameter	
xz. Root hair branching	
ya. Root hair density	
yb. Root hair length	
yc. Root hair diameter	
yd. Root hair branching	
ye. Root hair density	
yf. Root hair length	
yg. Root hair diameter	
yh. Root hair branching	
yi. Root hair density	
yj. Root hair length	
yk. Root hair diameter	
yl. Root hair branching	
ym. Root hair density	
yn. Root hair length	
yo. Root hair diameter	
yp. Root hair branching	
yq. Root hair density	
yr. Root hair length	
ys. Root hair diameter	
yt. Root hair branching	
yu. Root hair density	
yv. Root hair length	
yw. Root hair diameter	
yx. Root hair branching	
yy. Root hair density	
yz. Root hair length	
za. Root hair diameter	
zb. Root hair branching	
zc. Root hair density	
zd. Root hair length	
ze. Root hair diameter	
zf. Root hair branching	
zg. Root hair density	
zh. Root hair length	
zi. Root hair diameter	
zj. Root hair branching	
zk. Root hair density	
zl. Root hair length	
zm. Root hair diameter	
zn. Root hair branching	
zo. Root hair density	
zp. Root hair length	
zq. Root hair diameter	
zr. Root hair branching	
zs. Root hair density	
zt. Root hair length	
zu. Root hair diameter	
zv. Root hair branching	
zw. Root hair density	
zx. Root hair length	
zy. Root hair diameter	
zz. Root hair branching	

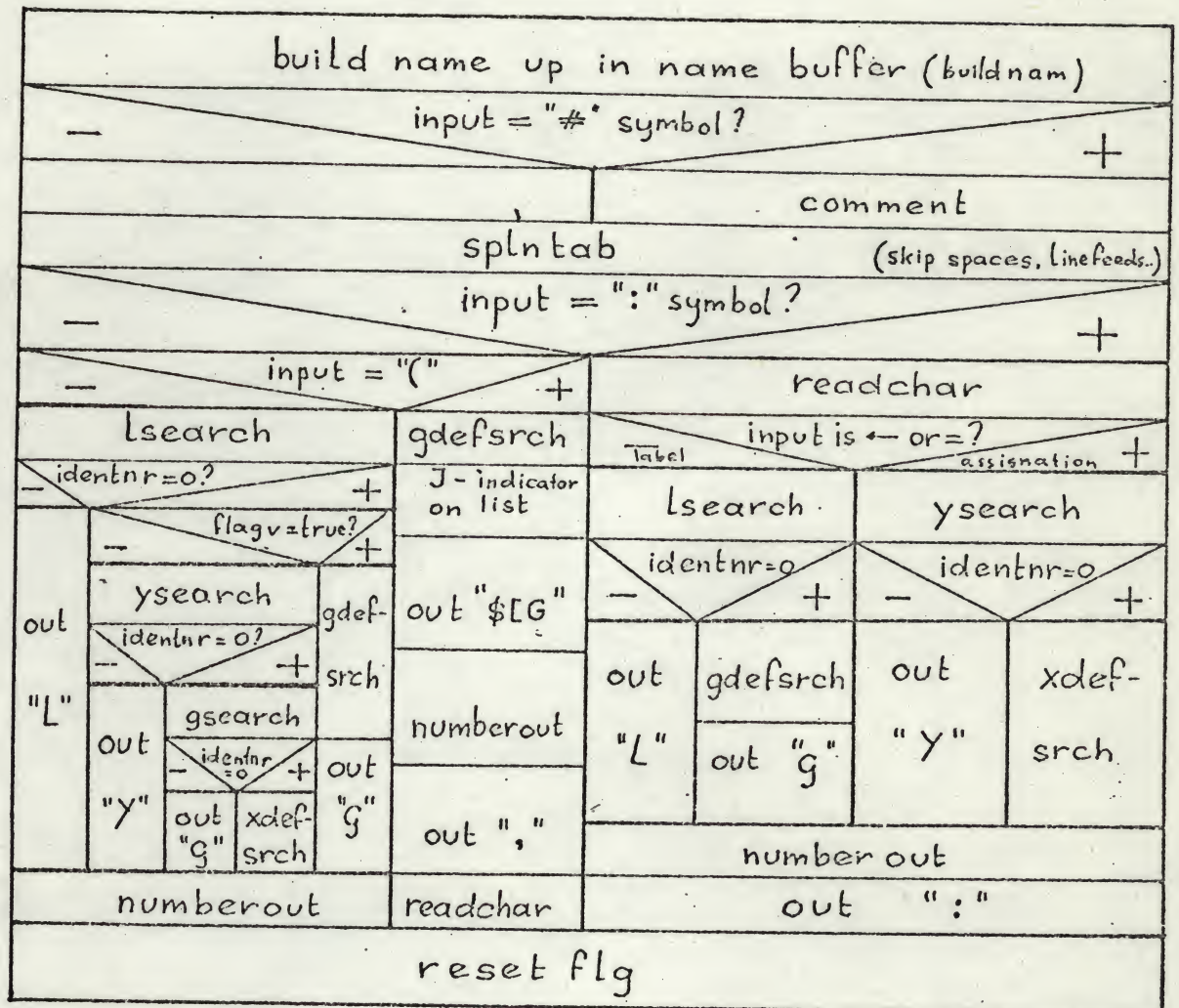
-basic symbols-

read midial basic symbol (name)											
lookup in symbol 'action table											
V	IF DO CASE BEGIN	FI OD ESAC END	case basic symbol equal to								not in table
GOTO			LABEL	GLOBAL	VECTOR	ARRAY	LOCAL FORMAL	VEC	EXPR SUBR	ELIF	
reset- flag	push- scope	pop- scope	List	List	List	List	List] -ind. on list		elifcnt	
vflag := true	printwd	od? - +	(2) -L-	(3) -G-	(4) -X-	(5) -X-	(1)			+ := 1	printwd
printwd	do? - +	printwd out "("					out "DCL(" -ypar")	out "["	proc	out "else" "if"	
							readchar				
resetflag											

LIST (arg)

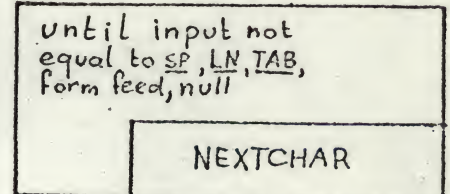
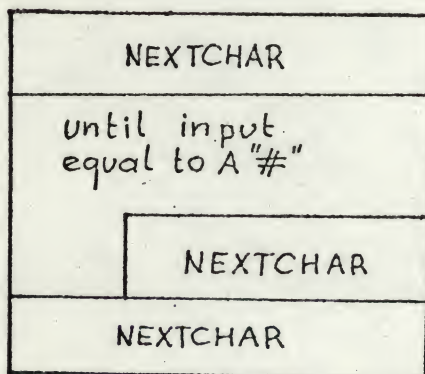
cond := true											
while cond											
splntab (skip spaces, linefeeds, tab)											
input = # symbol ?											
comment											
input = ; symbol ?											
input = alphabetical or numerical character ?											
cond := false											
read name into buffer (name)											
argument											
scope dependent											
scope independent											
read-char											
= 1											
= 2											
= 3											
= 4											
= 5											
xdef											
ydef											
Ldef											
gdef											
xdef											
read number											
xent += number											
read 1 character (readchar)											

NAME



COMMENT

SPLNTAB

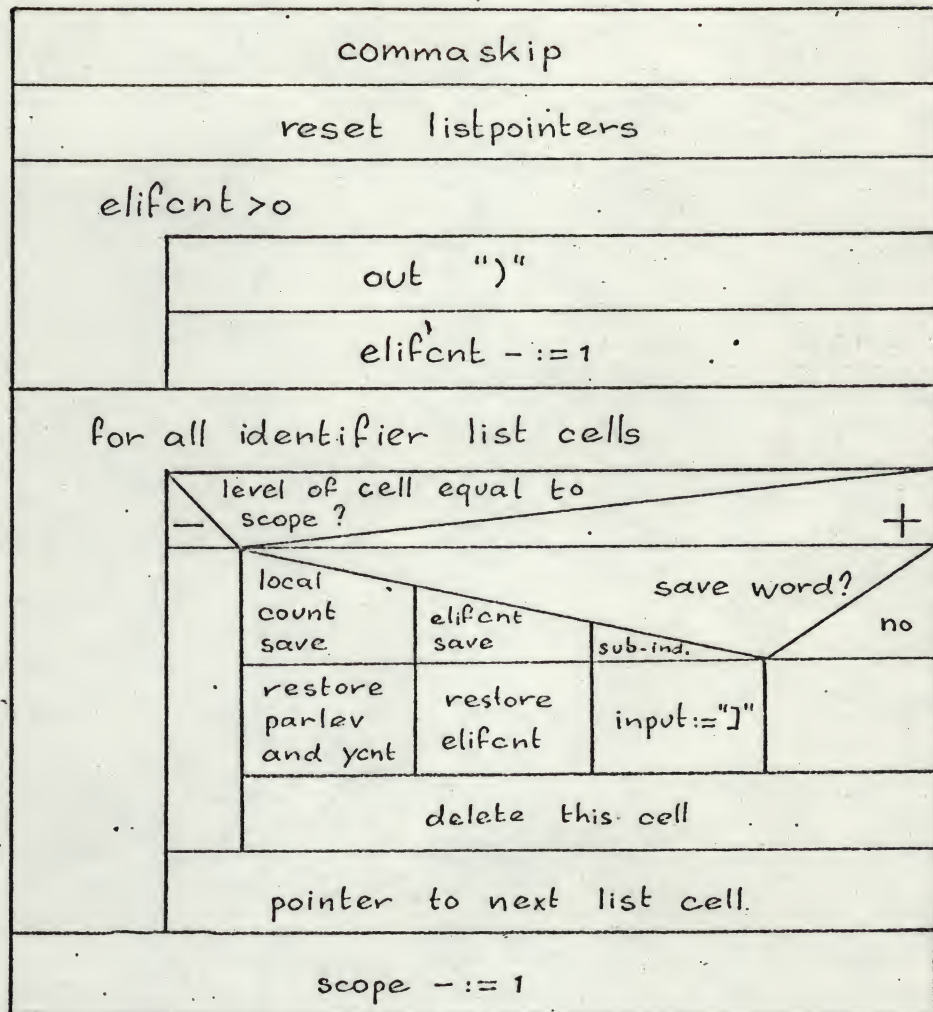


RESETFLG

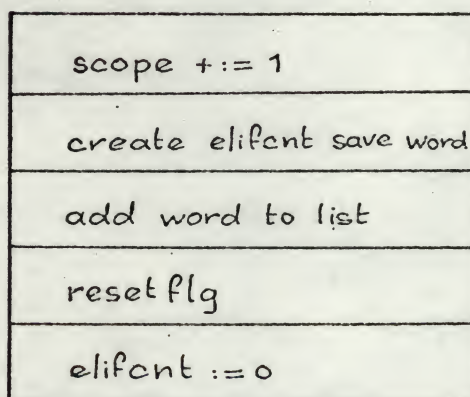
NUMBEROUT

flgcomma :=
flagv := false

output identnr
as decimal
number



pushscope



proc

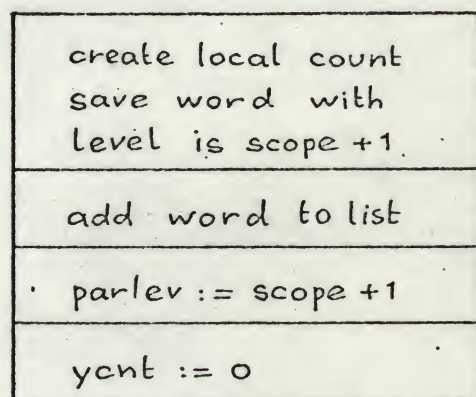




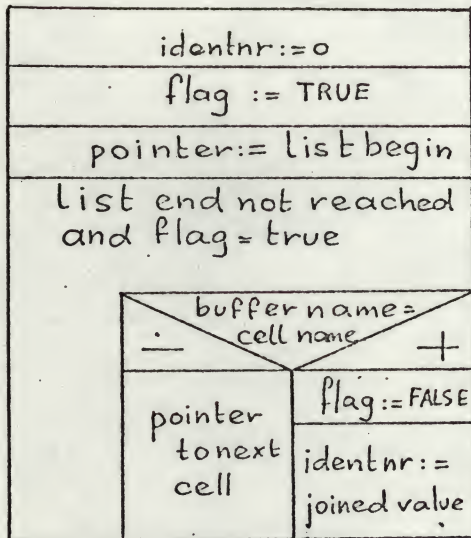
Figure 1



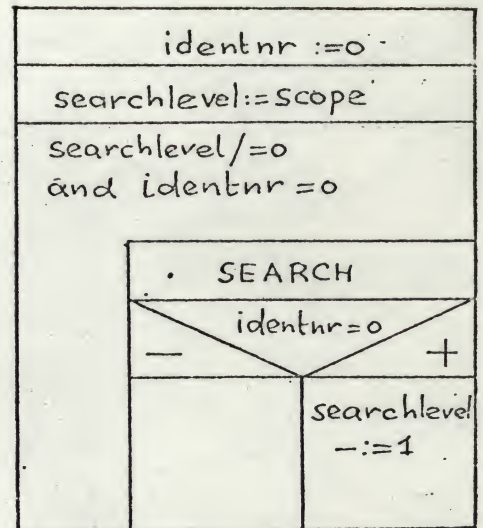
Figure 2



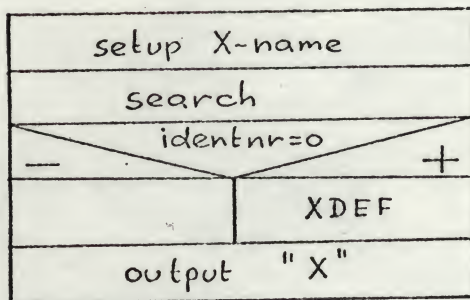
SEARCH



loops.EARCH



XDEF SRCH



GDEF SRCH

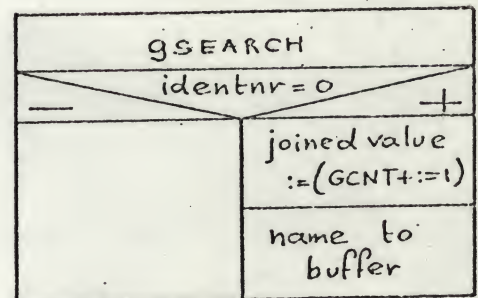


Figure 1



Figure 2



Figure 3



Figure 4



BEGIN #MIDIAL TO MINIAL PREPROCESSOR, VERSION 2
 DATE 75/01/14
 BY H. LUIIJF
 T.H. DELFT

LIST OF VARIABLES :

X1 -COMMON VARIABLE
 X2 -LAST INPUT CHARACTER
 X3 -INPUT CHARACTER CLASS
 X4 -G-COUNT
 X5 -X-COUNT
 X6 -L-COUNT
 X7 -Y-COUNT
 X8 -NAME LIST BEGIN POINTER
 X9 -NAME LIST END POINTER
 X10 -NAME LIST CURRENT POINTER FOR LOOKUP OPERATIONS
 X11 -CURRENT LEVEL
 X12 -NAME BUFFER : LEVEL AND CLASS
 X13 -FIRST TWO CHARACTERS OF NAME
 X14 -CHARACTERS 3 AND 4
 X15 -CHARACTERS 5 AND 6
 X16 -CHAIN NUMBER OF NAME
 X17 -CURRENT POINTER FOR SET ON LIST OPERATIONS
 X18 -COMMON VARIABLE
 X19 -FLAG FOR PREVENTING THE)) STRUCTURE: CONVERT TO ;SKIP)
 X20 -FLAG FOUND IN TABLE
 X21 -COMMON VARIABLE
 X22 -MIDIAL SPECIAL WORD CHAIN NUMBER
 X23 -NUMBER OF ELEMENTS
 X24 -CHARACTER 1
 X25 -CHARACTER 2
 X26 -CHARACTER 3
 X27 -CHARACTER 4
 X28 -CHARACTER 5
 X29 -CHARACTER 6
 X30 -FLAG V DETECTED.
 X31 -ELIF COUNT
 X32 -LOCAL VARIABLE LEVEL.

PROGRAM DRIVER

```
BEGIN #INITIALISE#
    DCL(AW0-AX0-B0060);
    P3:="(DCL(100));" ;
    X7:=X11:=X20:=X31:=0;X32:=1;
    X9:= (X8:=X10:=X17:=AX33 #SET AFTER X-LIST# )+5 ;
    CX10:=-1 ; #CREATE 1 EMPTY LIST CELL#
    VG34 ; #RESET FLAGS#
    X4:=X5:=X6:=20 ; #RESET X , L AND G COUNT#
    VG3; #READ AND CLASSIFY#
    #TEXT SCREENING ON THE HIGHEST LEVEL#

    WHILE X3/=3 DO
        BEGIN
            IF X3=1 THEN VG40 #IDENTIFIER/LABEL# ELSE
            IF X3=4 THEN VG1 #MIDIAL SYMBOL# ELSE
```

RECEIVED
JAN 14 1971
FBI - NEW YORK

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70

RECEIVED 12/11/70


```

IF X3=7 THEN VG11 #COMMENT DETECTED# ELSE
IF X3=14 THEN VG31 #SP, LN, LF# ELSE
  IF X3=5 THEN VG6 #OPEN PARENTHESIS# ELSE
  IF X3=6 THEN VG7 #CLOSING PARENTHESIS DETECTED,
    TEST PREVIOUS CHAR=; ,POP ,READ # ELSE
  IF X3=12 THEN VG34;VG5 #% DETECTED
    RESET X19;OUT % AND FOLLOWING CHAR# ELSE
  IF X3=8 THEN X19:=TRUE #) DETECTED# ELSE
  IF X3=13 THEN VG34;VG5; WHILE X3/=13 DO VG5
    #SKIP " " STRING# ELSE ;
    VG34 #OUTPUT CHARACTER#
  FI FI FI FI FI ; VG5 #PRINT AND READ 1 CHARACTER#
  FI FI FI FI
END ;
VG35; #TEST ;#

```

```

#OUTPUT MISSING CLOSING PARENTHESIS#
IF X11>0 THEN P1+="#".DECS(X11)." ) MISSING0" ;
  WHILE X11>0 DO
  BEGIN VG7 ;#POP AND DELETE LEVEL#
    VG8 #PRINT A ) SYMBOL#
  END

```

```

FI;
P3*=[X], X; , LN , LN , B3232) #OUTPUT THE ) OF THE (DCL....#
END ;

```

-----#

EXPR(G1: #MIDIAL TO MINIAL STANDARD WORDS CONVERTING#

```

#READ MIDIAL SYMBOL STRING AND SEARCH IN TABLE#
VG3 ; #READ SYMBOL#
VG14 ; #READ SYMBOL STRING#
X1:=G50 ; VG12 #X18:=TRUE# ;
WHILE (X1<=G51)&X18 DO
BEGIN
  X22:=(IF (X1 HD AX23)&(AX23 HD X1) THEN #IN TABLE#
    VG13 #X18:=FALSE# ; C(X1+CX1+1)
    ELSE X1+=CX1+2 ; 0
  FI
);RESET
END ;

```

```

IF (X22&B0010)=B0010 THEN
BEGIN #FI 00 ESAC END#
  #IF PREVIOUS=; THEN OUT ;SKIP #
  VG7 ; #POP LEVEL AND DELETE#
  IF X22=B0012 THEN #OD# VG8 #OUTPUT ) SYMBOL#
    ELSE VG2 #OUTPUT MINIAL SYMBOL#
  .FI

```

```

END
ELSE
BEGIN
  VG34 ; #RESET X19#
  IF X22=0 THEN #NOT FOUND# VG2 #OUTPUT SYMBOL#
  ELSE
    IF (X22&B7770)=0 THEN
    BEGIN
      VG6 ; #PUSH LEVEL#
      VG2; #OUTPUT SYMBOL#
      IF X22=B0002 THEN #DO# P3:=%( FI

```

THE
OFFICE OF THE
SECRETARY OF THE
NAVY
WASHINGTON, D. C.
JANUARY 1, 1900

TO THE
HONORABLE
MEMBERS OF THE
NAVY
DEPARTMENT
WASHINGTON, D. C.

THE
OFFICE OF THE
SECRETARY OF THE
NAVY
WASHINGTON, D. C.
JANUARY 1, 1900

THE
OFFICE OF THE
SECRETARY OF THE
NAVY
WASHINGTON, D. C.
JANUARY 1, 1900


```

END
ELSE
  IF X22=B0401 THEN #GOTO# X30:=TRUE;VG2 ELSE
  IF X22=B2000 THEN VG2;X14:=X32; X13:=X7;
  #SUBR/EXPR LOCAL COUNT CHANGE# X12:=B6000+(X32:=X11+1);
  VG9 ; #TO LIST# X7:=0 ELSE
  IF X22=B0021 THEN #VECTOR LIST# $(G42,4)
  ELSE
  IF X22=B0022 THEN #ARRAY# $(G42,5) ELSE
  IF X22=B0041 THEN #GLOBAL LIST# $(G42,3)
  ELSE
  IF X22=B0042 THEN #LABEL LIST# $(G42,2)
  ELSE
  IF X22=B1000 THEN #ELIF# X31+:=1;P3*="ELSE IF"
  ELSE
  IF X22=B0100 THEN
  #FORMAL OR LOCAL LIST# $(G42,1) ;
  P3*="DCL(", DECS(X7) ,");"
  ELSE
  IF X22=B0201 THEN #VECH
  VG20; #PUSH AND SPECIAL#
  X2:=X1
  ELSE
  IF X22=B0400 THEN #V# (X30:=TRUE;X2:=%V)
  FI
  FI ;
  VG5
  FI
  FI
  FI
  FI
  FI
  FI
  FI
  FI
  FI
  END
FI

```

)

EXPR(G2: P3*AX23 #OUTPUT THE MINIALGOL STANDARD WORD#);

```

EXPR(G3: #READ AND CLASSIFY INPUT CHARACTER#
X3:=( IF ((X2:=R3) >=%A)&(X2<=%Z) THEN #ALPHABETIC# 1 ELSE
IF (X2 >=%0)&(X2<=%9) THEN #NUMERIC# 2 ELSE
IF X2=B0232 THEN #CNTRL Z# 3 ELSE
IF X2=%' THEN #IDIAL SYMBOL# 4 ELSE
IF X2=%( THEN #OPENING PARENTHESIS# 5 ELSE
IF X2=%) THEN #CLOSING PARENTHESIS# 6 ELSE
IF X2=%# THEN #COMMENT DETECTED# 7 ELSE
IF X2=%: THEN #END OF LIST OR STATEMENT# 8 ELSE
IF X2=%: THEN #LABEL DECLARED# 10 ELSE
IF (X2=%=)! (X2=%0) THEN #ASSIGN# 11 ELSE
IF X2=%X THEN #CHARACTER VALUE# 12 ELSE
IF X2=%" THEN # " " STRING# 13 ELSE
IF (X2=SP)! (X2=0)! (X2=B211)! (X2=B212)! (X2=LN)! (X2=B200)
!(X2=B214) THEN #SPACE,TAB,LINE FEED,CARR.RETURN# 14
ELSE 20 FI FI FI FI FI FI FI FI FI FI FI FI FI FI FI))

```

EXPR(G4: #PRINT ONE CHARACTER# P3:=X2);

THE UNIVERSITY OF CHICAGO

DEPARTMENT OF CHEMISTRY

1100 SOUTH EAST ASIAN AVENUE

CHICAGO, ILLINOIS 60607

TEL: (312) 937-1234 FAX: (312) 937-1234

WWW.CHEM.UCHICAGO.EDU

OFFICE OF THE DEAN OF CHEMISTRY

1100 SOUTH EAST ASIAN AVENUE

CHICAGO, ILLINOIS 60607

TEL: (312) 937-1234 FAX: (312) 937-1234

WWW.CHEM.UCHICAGO.EDU

OFFICE OF THE DEAN OF CHEMISTRY

1100 SOUTH EAST ASIAN AVENUE

CHICAGO, ILLINOIS 60607

TEL: (312) 937-1234 FAX: (312) 937-1234

WWW.CHEM.UCHICAGO.EDU

OFFICE OF THE DEAN OF CHEMISTRY

1100 SOUTH EAST ASIAN AVENUE

THE UNIVERSITY OF CHICAGO

DEPARTMENT OF CHEMISTRY

1100 SOUTH EAST ASIAN AVENUE

CHICAGO, ILLINOIS 60607

TEL: (312) 937-1234 FAX: (312) 937-1234

WWW.CHEM.UCHICAGO.EDU

OFFICE OF THE DEAN OF CHEMISTRY

1100 SOUTH EAST ASIAN AVENUE

CHICAGO, ILLINOIS 60607

TEL: (312) 937-1234 FAX: (312) 937-1234

WWW.CHEM.UCHICAGO.EDU

OFFICE OF THE DEAN OF CHEMISTRY

1100 SOUTH EAST ASIAN AVENUE

CHICAGO, ILLINOIS 60607

TEL: (312) 937-1234 FAX: (312) 937-1234

WWW.CHEM.UCHICAGO.EDU

OFFICE OF THE DEAN OF CHEMISTRY

EXPR(G5: #PRINT, READ, CLASSIFY# VG4; VG3);

-87-

EXPR(G6: #PUSH LEVEL# X12:=B5000+(X11+:=1); Y13:=X31;
#STORE ELIF COUNT; AND RESET FLAGS#
VG9; VG34 ; X31:=0);




```

EXPR(G7:  #POP LEVEL#
          #TEST ; FLAG#   VG35 ;
          #DELETE FROM LIST# X10:=X8;
          WHILE X31>0 DO (X31-:=1;VG8 #CLOSE ELIF# ))
          WHILE X10<X9 DO
          BEGIN IF ((X1:=CX10)=B6000+X11)!(X1=B6000+X11+1)
                THEN CX10:=-1; X7:=C(X10+1) ; X32:=C(X10+2)
                #RESTORE Y COUNT AND PARAMETER LEVEL#
                ELSE IF X1&B0777=X11 #SAME LEVEL# THEN
                  X1-:=X11 ;
                  IF X1=B4000 THEN #SUB CHAR.#
                    X2:=X1
                  ELSE IF X1=B5000 THEN X31:=C(X10+1)
                  #RESTORE ELIF COUNT#
                  FI
                FI ; CX10:=-1 #DELETE CELL#
                FI
          FI;
          X10+:=5 #NEXT NAME#
        END;
        X11-:=1; #LEVEL-:=1#
        X17:=X10:=X8 #RESET LIST POINTERS# ))

EXPR(G8:  P3:=X) #OUTPUT A ) SYMBOL#);

EXPR(G9:  #SET NAME ON LIST#   VG12 #X18:=TRUE#)
          WHILE X18 DO
          BEGIN IF X17<X9 THEN
                IF CX17=-1 THEN VG10 #COPY IN LIST#
                ELSE X17+:=5
                FI
                ELSE X9+:=5;VG10 #COPY AFTER LIST#
                FI
          END
          ))

EXPR(G10:  #COPY NAME FROM BUFFER TO LIST CELL#
          X21:=AX12-1;X1:=X17-1;
          WHILE (X21+:=1)<AX17 DO
            C(X1+:=1):=CX21 ;
          VG13 #X18:=FALSE#);

EXPR(G11:  #SKIP COMMENTS#   VG5;WHILE X3/=7 DO VG5;VG5);

EXPR(G12:  X18:=TRUE );

EXPR(G13:  X18:=FALSE );

EXPR(G14:  #NAME DETECTED;SETUP BUFFER#
          VG12 #X18:=TRUE#; X1:=X12:=X13:=X14:=X15:=X16:=
          X23:=X24:=X25:=X26:=X27:=X28:=X29:=0 ;
          WHILE (X1&8(X1<6)) DO
            BEGIN IF (X3=1)!(X3=2) THEN
                  BEGIN C(AX24+X1):=X2 ;
                  X2-:=B0257;#BUILD NAME#
                  C(AX13+(X1/2))+:=(IF (X1-((X1/2)*2))=0 THEN
                                      X206 ELSE X2 FI);
                  VG3;#READ AND CLASSIFY#
                  X23:=(X1+:=1) #NAME LENGTH#
                END
            ELSE VG13 #X18:=FALSE#
            FI
          )

```



```

END;
WHILE X18*((X3=1)|(X3=2)) DO
  BEGIN VG3  #NAME LONGER THAN 6 CHAR,SH
  END
  )

```

```

EXPR(G20: #CREATE SPECIAL: AT LEVEL=POP THEN ) TO )#
VG6;#PUSH LEVEL# X12:=B4000+X11/VG9#TO LIST# );

```

```

EXPR(G21: #LOOKUP IN TABLE#
  X10:=X8-5;X16:=0;X27:=TRUE;
  WHILE (X27*((X10+:=5)<X9)) DO
  BEGIN X1:=-1;VG12 #X18:=TRUE#;
    WHILE (X18*((X1+:=1)<4)) DO
      IF C(AX12+X1)/=C(X10+Y1) THEN VG13 # X18:=FALSE# FI;
      IF X18 THEN X20:=FALSE #FOUND#;X16:=C(X10+4)
      FI #NOT FOUND: LOOP#
    END
  )

```

```

EXPR(G22: #SEARCH X-NAME# X12:=B3000/VG21; #SEARCH X#
  IF X16=0 THEN #A NEW NAME#
  V( EXPR (G221: X12:=B3000 ; X16:=(X5+:=1);#CHAIN#
    VG9 #ADD TO LIST# )
  FI; XX #OUTPUT X# )

```

```

EXPR(G23: #SEARCH G#
  X12:=R2000;#SETUP G# VG21 #AND SEARCH# )

```

```

EXPR(G24: #SEARCH LOOP UNTIL 0 LEVEL#
  X12+:=1; X16:=0 ;
  WHILE (((X12-:=1)&B0777)/=0) & (X16=0)) DO
    VG21 #SEARCH#
  )

```

```

EXPR(G25: #SEARCH Y IDENTIFIER#
  V EXPR(G251: #Y IDENTIFIER# X12:=X32+B1000 #LEVEL X32#);
  VG24 #SEARCH# )

```

```

EXPR(G26: #SEARCH L-LABEL#
  V EXPR(G261: #L LABEL# X12:=X32 #LEVEL X32#);
  VG24 #SEARCH# )

```

```

EXPR(G27: #OUTPUT X16# P3*=(DECS(X16)).(SP) )

```

```

EXPR(G28: #SEARCH AND DEFINE G# VG23 ; #SEARCH G#
  IF X16=0 THEN X16:=(X4+:=1);VG9 FI );

```

```

EXPR(G29: #SEAPCH AND DEFINE Y# VG25 ;
  IF X16=0 THEN
    V EXPR (G291: VG251 ; X16:=(X7+:=1) ; VG9 )
  FI

```

```

EXPR(G30: #DEFINE L-LABEL#
  VG261 ; #SETUP L# X16:=(X6+:=1) ; VG9 #TO LIST# );

```

```

EXPR(G31: #SKIP AND OUTPUT CR,LF,TAB,SP#
  WHILE X3=14 DO VG5
  )

```

```

EXPR(G34: #RESET X19 FLAG# X19:=X30:=FALSE )

```

```

EXPR(G35: #TEST ) COMBINATION;OUTPUT "SKIP" IF SO#
  IF X19 THEN P3*="SKIP ";VG34#RESET# FI );

```

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
CHICAGO, ILLINOIS 60637

TO: THE DIRECTOR, NATIONAL BUREAU OF STANDARDS
WASHINGTON, D.C. 20535

FROM: DR. J. H. GOLDSTEIN, CHAIRMAN
COMMISSION ON THE MEASUREMENT OF
THE SPEED OF LIGHT

SUBJECT: REPORT ON THE MEASUREMENT OF THE
SPEED OF LIGHT IN VACUUM

1. INTRODUCTION

The purpose of this report is to present the results of the measurements of the speed of light in vacuum, as conducted by the Commission on the Measurement of the Speed of Light, under the leadership of Dr. J. H. Goldstein.

2. SUMMARY OF RESULTS

The measurements were conducted using a variety of methods, including the use of a rotating mirror, a modulated light source, and a frequency-stabilized laser. The results of these measurements are summarized in the following table:

Method	Value of c (m/s)	Uncertainty (m/s)
Rotating mirror	299,792,458	± 10
Modulated light source	299,792,458	± 10
Frequency-stabilized laser	299,792,458	± 10

3. CONCLUSIONS

The results of these measurements are in excellent agreement with the value of c as determined by the International Union of Pure and Applied Chemistry (IUPAC) in 1983.

4. REFERENCES

1. J. H. Goldstein, "The Speed of Light in Vacuum," *Phys. Rev. Lett.*, **45**, 1000 (1980).

2. J. H. Goldstein, "The Speed of Light in Vacuum," *Phys. Rev. Lett.*, **45**, 1000 (1980).

3. J. H. Goldstein, "The Speed of Light in Vacuum," *Phys. Rev. Lett.*, **45**, 1000 (1980).


```

EXPR(G40: #IDENTIFIER DETECTED#          VG14;#READ NAME#
IF X3=7 #COMMENT# THEN VG11 FI;
VG31 ; #SKIP SPACES,LINE FEEDS ETC#
IF X3=10 THEN 4: DETECTED#
BEGIN
    VG3; #READ NEXT#
    P3:= IF X3=11 #:=# THEN
        BEGIN
            VG25 ; #SEARCH Y#
            IF X16=0 THEN VG22 #X IDENTIFIER#
                ELSE XY #Y IDENTIFIER#
            FI
        END
        ELSE #LABEL#
            VG26; #SEARCH L#
            IF X16=0 THEN #G# VG28;XG #SEARCH AND DEFINE#
                ELSE XL
            FI
        FI ;
        VG27 ; P3:=X; #OUTPUT X16 AND :#
    END
ELSE
    IF X3=5 #X( # THEN #PROCEDURE NAME#
        VG28; #SEARCH AND DEFINE G#
        VG22; #SPECIAL TO LIST#
        P3*="S(G";VG27;Y2:=X;VG3
        #OUTPUT S(G....,#
    ELSE
        P3:= ( VG26; #LOCAL LABEL?#
        IF X16=0 THEN
            IF X30 THEN VG28 ; XG #FUNCTION#
            ELSE
                BEGIN VG25 ; #SEARCH Y#
                    IF X16=0 THEN
                        BEGIN VG23 ; #SEARCH G#
                            IF X16=0 THEN VG22 #SEARCH AND DEF X#
                                ELSE XG
                            FI
                        END
                    ELSE XY
                    FI
                END
            FI
        ELSE XL
        FI ) ;
        VG27 #PRINT NUMBER#
    FI
    ;VG34 #RESET X19 AND X30# ))

```

```

SUBR(G42: #LIST OF NAMES# Y22:=TRUE;
  WHILE X22 DO
    BEGIN
      VG31: #SKIP LF,CR,TAB,SP#
      IF X3=8 THEN X22:=FALSE #/ IS LIST END#
      ELSE IF X3=1 THEN
        VG14: #READ NAME#
        CASE Y1 IN
          #Y# VG291 ,
          #L# VG30 ,
          #G# VG28 ,
          #X# VG221, #VECTOR LIST#
          #X# BEGIN VG221: #SET XNAME#
            X1:=0; WHILE X3/=2 DO VG3 ;
            #READ UNTIL NUMERIC#
            WHILE X3=2 DO (X1:=(X1*10)+(X2-X0);VG3);
            X5+:X1
          END
        ESAC
      ELSE VG3
      FI
    FI
  END ; VG3
)
```

```

TABLE(G50:
  2,XI,XF,B0001,
  2,XD,XD,B0002,
  4,XC,XA,XS,XE,B0003,
  5,XB,XE,XG,XI,XN,B0004,

  2,XF,XI,B0011,
  2,XD,XD,B0012,
  4,XE,XS,XA,XC,B0013,
  3,XE,XN,XD,B0014,

  6,XV,XE,XC,XT,XO,XR,B0021,
  5,XA,XR,XR,XA,XY,B0022,

  6,XG,XL,XO,XB,XA,XL,B0041,
  5,XL,XA,XB,XE,XL,B0042,

  6,XF,XO,XR,XM,XA,XL,B0100,
  5,XL,XO,XC,XA,XL,B0100,

  3,XV,XE,XC,B0201,

  1,XV,B0400,
  4,XG,XO,XT,XO,B0401,

  4,XE,XL,XI,XF,B1000,

  4,XS,XU,XB,XR,B2000,
  4,XE,XX,XP,XR,B2000
)
G51:
END;
```

1901-1902

1902-1903

1903-1904

1904-1905

1905-1906

1906-1907

1907-1908

1908-1909

6.5 THE MIDIAL SYSTEM.

As stated before, the Midial system consists out of two save systems, a preprocessor system and a compiling and run time system.

The preprocessor system starts with the Command Decoder call: the input and output files must be given to the system.

Thereafter, if there is a program specified, the preprocessor translates the midial program to an intermediate minial program.

This intermediate program file can be named in the file command string, or is created by default as MIDIAL.MT on the SYS (-tem) device.

After preprocessing, the midial system looks after the /P option switch. If it was specified, the system chains to a new midial preprocessor system, if it was not specified, the midial preprocessor system chains to a compiler and runtime system. (MIDICM.SV)

From there a slightly modified Minial system is used. Only the possibility of interrupting by typing in the +C (control C) character on the systems keyboard is new.

After a normal return out of the program, the Midicm system test the /M option switch . If it was specified the system returns to the Monitor, else it returns to the Midial preprocessor system. (MIDIAL.SV)

It is possible to load a new preprocessor (minial binary) over an old Midial system by starting this system on the second start point, the load entry. (B4000)

After the loading, the system returns to the monitor. From there the new system can be saved with the SAVE command. (see also page 95)

The MIDIAL.SV and MIDICM.SV can be saved under other file names, after modifying the chain name characters.

They are packed in the locations 00060 till 00062₈ .

On the next two pages flowcharts sre given of the MIDIAL and MIDICM systems.

Load entry.

command decoder call
lsw := 0
lookup preprocessor binary file
replace the program file information by the binary file information

Run-time entry.

command decoder call	
lsw := 1	
+	program file present?
	-
+	intermediate program file specified?
	-
-	extension blank?
+	
	extension := .MT
	name becomes MIDIAL.MT
open the input file	
fetch the input handler	
read 1 input block into the input buffer	
swap the Loader and the monitor (swsw:=1)	
switch the interrupt enable on	
lsw = 0?	
+	-
load the preprocessor	run the preprocessor (start at ruarg)
interrupt enable off	interrupt enable off
swap (swsw:=0)	swap (swsw:=0)
save startpoint for the runsystem at ruarg	close the output file

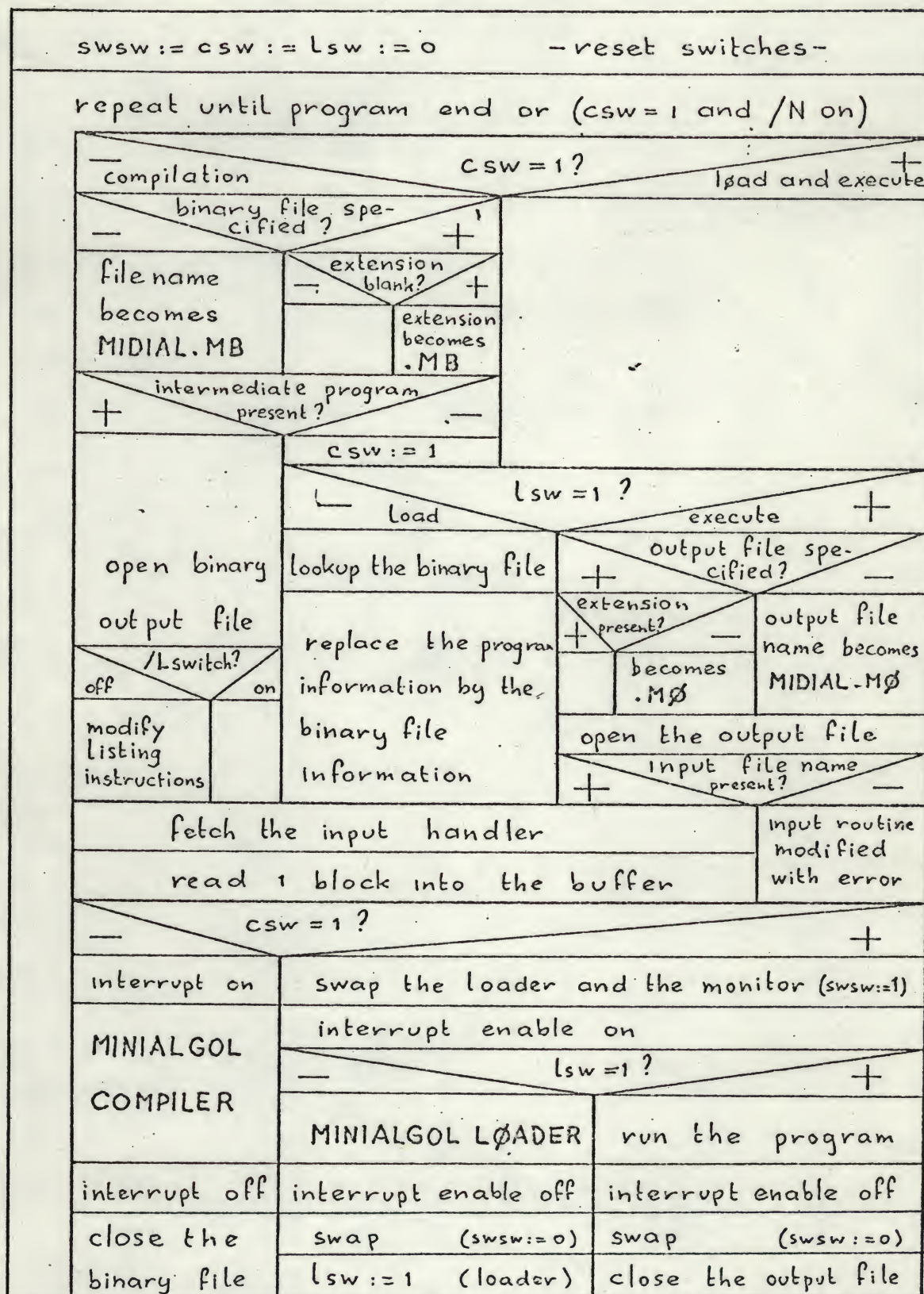
MIDIALGOL system

while not interrupted or /M switch on	
MIDIAL.SV	
-preprocessor system-	
+	/P switch on ?
	-
	MIDICM.SV
	-compiler and runtime system-
return to the monitor	

<p>1. The first part of the report is a general introduction to the subject of the study. It discusses the importance of the problem and the objectives of the research.</p> <p>2. The second part of the report is a detailed description of the methods used in the study. It includes a description of the experimental design, the data collection procedures, and the statistical methods used for data analysis.</p> <p>3. The third part of the report is a presentation of the results of the study. It includes a description of the data, a discussion of the findings, and a comparison of the results with previous research.</p> <p>4. The fourth part of the report is a conclusion and a discussion of the implications of the study. It includes a summary of the findings, a discussion of the limitations of the study, and suggestions for future research.</p>	<p>5. The fifth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p> <p>6. The sixth part of the report is an appendix. It includes a list of the tables, figures, and other materials used in the study.</p> <p>7. The seventh part of the report is a glossary. It includes a list of the terms used in the study and their definitions.</p> <p>8. The eighth part of the report is a bibliography. It includes a list of the books, articles, and other sources used in the study.</p>
<p>9. The ninth part of the report is a list of tables. It includes a list of the tables used in the study and their descriptions.</p> <p>10. The tenth part of the report is a list of figures. It includes a list of the figures used in the study and their descriptions.</p> <p>11. The eleventh part of the report is a list of other materials. It includes a list of the other materials used in the study and their descriptions.</p> <p>12. The twelfth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p>	<p>13. The thirteenth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p> <p>14. The fourteenth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p> <p>15. The fifteenth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p> <p>16. The sixteenth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p>

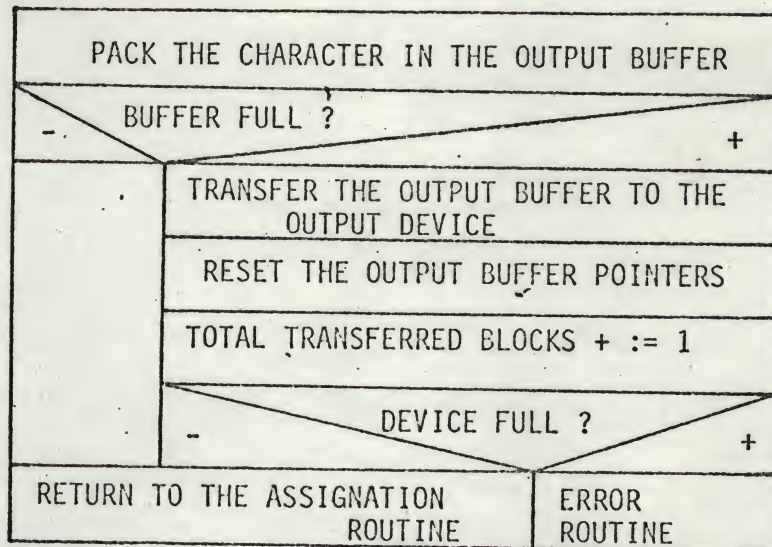
Page 10 of 10

<p>17. The seventeenth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p> <p>18. The eighteenth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p> <p>19. The nineteenth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p> <p>20. The twentieth part of the report is a list of references. It includes a list of the books, articles, and other sources used in the study.</p>

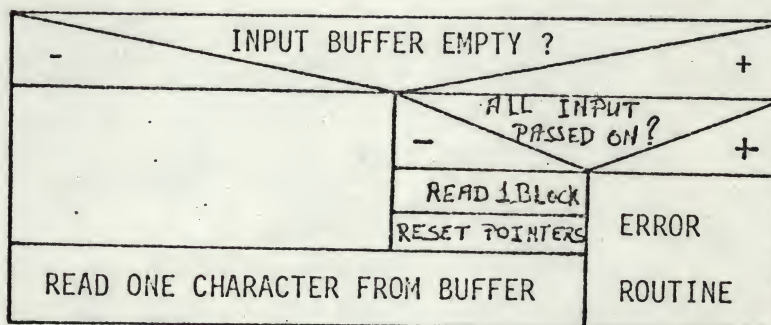


THE INPUT AND OUTPUT CHANNELS 3 AND 4 .

OUTPUT



INPUT



6.6 SYSTEM CREATION FROM BINARIES.

For creating a new MIDIAL system, the programmer needs the binary file of the MIDIAL system (MIDISY.BN) , and the Minial binary of the preprocessor. The following command string must be typed in :

```
.RU dev ABSLDR. +  
*dev : MIDISY.BN$  
.ST 4000 +  
*dev : PREMIN.MB < +  
.SAVE dev MIDIAL 0-177,4000-5577,10000-17577 ; 4200 =0 +
```

For creating the MIDIAL compiler system (MIDICM) , the programmer needs the compiler system binary (MIDCSY.BN) and the compiler binary (MIDCCM.BN) . The command string is as following :

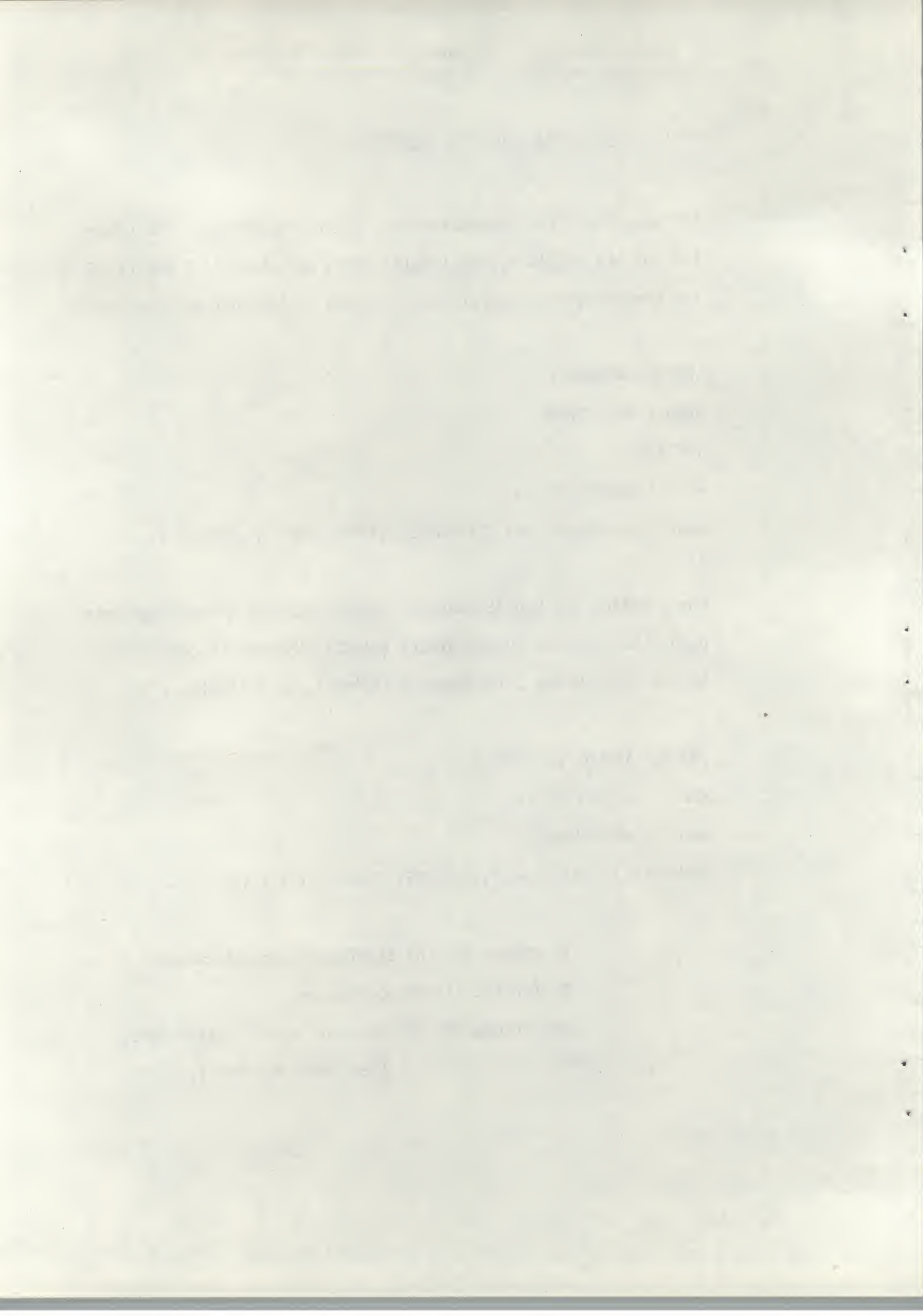
```
.RU dev. ABSLDR. +  
*dev : MIDCSY.BN +  
*dev : MIDCCM.BN $  
.SAVE dev MIDICM 0-177,4200-5577,10000-17177 ; 4200 =0 +
```

+ stands for the carriage-return character,

\$ for the altmode character.

dev stands for the input or output device code.

(see PDP-8 handbooks)



6.7 THE FILE COMMAND STRING FOR THE MIDIALGOL SYSTEM.

The midialgol system asks the user to identify which files must be used by the preprocessor, the compiler and the load-run system. The user can also give some control information to the system to do only a part of the whole preprocess, compile, load and go system process.

The command string must be typed in by the user after the * of the Command Decoder :

```
*{ {<dev>:}{BINFILE{.ex1}} {,{<dev>:}{OUTFILE{.ex2}} {,{<dev>:}{PREPROG{.ex3}} <- }
    { {<dev>:}{PROGRAM{.ex4}} } {,{<dev>:}{INFILE{.ex4}} } {/L}{/M}{/N}{/P}
```

{.....}

- means that this sequence can be omitted, and that in some cases this sequence will be replaced by a default sequence if it was omitted.

{<dev>:}

- <dev> stands for the device code word.
(e.g. DSK , P , R)
- if the device code is not specified, SYS is taken as the default device.

{BINFILE{.ex1}}

- BINFILE.EX is the file in which the compilation result is set.(binary file)
This file is also the file that will be loaded by the loader.

THE HISTORY OF THE

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

- If the `BINFILE` is not specified by the user , the default name of the file is : `MIDIAL.MB` .

`{.ex1}`

`{OUTFILE{.ex2}}`

- If the extension is not specified , it will have the default value of `.MB` .
- `OUTFILE.EX` is the file name of the output file at run time on which the output data is written by the 'P3 and 'P4 instructions of Midialgol.

- If the file name is not specified, then the default file `MIDIAL.MO` is generated if there has been output on 'P3 or 'P4 .

`{.ex2}`

- If the extension is not specified by the user it gets the value of `.MO` .

`{PREPROG{.ex3}}`

- `PREPROG.EX` is the output file of the preprocessor. This file becomes the input file for the compiler.

On this place old Minialgol program files can be set for compiling.

- If the file name is not specified and there is a `PROGRAM`-file present, the file gets the default name of `MIDIAL.MT` .
- If there is no `PROGRAM`-file specified, and the `PREPROG`-file is specified then the `PREPROG`-file is compiled and eventually runned.
- The default extension for this file is `.MT`.

`{.ex3}`



{<}

- is the delimiter between the output- and the input files. If there are no output files to specify for running a program, the user can omit this sign.

Instead of the < symbol the + symbol can be used.

{PROGRAM{.ex4}}

- PROGRAM.EX is the name of the file in which the Midialgol program is put away.

{INFILE{.ex4}}

- INFILE.EX is the file name of the file from which the data is obtained by the 'R3 and 'R4 instructions in Midialgol.
- If the device code and the filename are not specified, the system returns after an 'R3 or 'R4 instruction to the Monitor, with an error printout. ("I/O ERR AT nnnn")

{.ex4}

- The default value for the extension of the input files is .MI (.Midial Input)
- If the user does not specify an input file extension, the default value (.MI) is added to the file name. If this file is not found on the specified (or default) device, a search for a file without an extension is made.

{/L}

- /L is the switch by which the user can obtain the listing at compile time of the (preprocessed) program.

{/M}

- is specified in the case that the control of the system must return, after completion to the Monitor.

In the case that the file MIDIAL.SV is

not on the SYS (-tem) device, this switch option MUST be specified.

(Else you get an Monitor error)

- If this swich is not specified, the control returns to a fresh Midialgol system. This is done by chaining to the MIDIAL.SV file from the MIDICM.SV file. The system returns with the * of the Command Decoder.

{/N}

| only compilation

- This is the No load switch, to do only (a) compilation(s).
- This option can be used independently of the /M option.

{/P}

| only preprocess

- This switch is used for doing only the Preprocess fase. After completion the control of the system returns to the Command Decoder.(The /M switch does not work here.)



| only load-and go

- If there are no specifications for files on the places of the program and preprocessed file, the file on the *BINFILE*-place (or its default value) is loaded and runned.

{<devi>:}

- For *devi* apply normally the rules for *dev* , only in the case that there is no program specified the user MUST specify here a device code.

Note: In the case that an input- or output- device is a non-directory device (e.g. P or R) the file name can be omitted.

For output files the default name is generated, but this name is of no significance.

...the ...
 ...the ...
 ...the ...
 ...the ...
 ...the ...

...the ...
 ...the ...
 ...the ...
 ...the ...
 ...the ...

...the ...
 ...the ...
 ...the ...
 ...the ...
 ...the ...

...the ...
 ...the ...
 ...the ...
 ...the ...
 ...the ...

...the ...
 ...the ...
 ...the ...
 ...the ...
 ...the ...

/P	Prog	Preprog	Binary	/N	Output	preproc		compiler		load	run out
						in	out	in	out		
						PROGRAM	PREPROC	PREPWR	BINARY	BINARY	OUTPUT
										D	D
					1					D	F
			1							F	D
			1		1					F	F
		1						F	D	D	D
		1			1			F	D	D	F
		1		1	x			F	D		
		1	1					F	F	F	D
		1	1		1			F	F	F	F
		1	1	1	x			F	F		
	1					F	D	D	D	D	D
	1				1	F	D	D	D	D	F
	1			1	x	F	D	D	D		
	1		1			F	D	D	F	F	D
	1		1		1	F	D	D	F	F	F
	1		1	1	x	F	D	D	F		
	1	1				F	F	F	D	D	D
	1	1			1	F	F	F	D	D	F
	1	1		1	x	F	F	F	D		
	1	1	1			F	F	F	F	F	D
	1	1	1		1	F	F	F	F	F	F
	1	1	1	1	x	F	F	F	F		
1	1		x	x	x	F	D				
1	1	1	x	x	x	F	F				

	means not specified.
1	means that the file is specified.
x	means that specifying a file name here has no meaning.
F-the specified file is taken for the I/O.	
D-the default file is taken for the I/O.	

6.8 THE BINARY LOADER.

The loader of the Midialgol system is swapped by the control system with the Monitor page of field 1 .

The loader is present in the monitor page at the moment that a program is loaded, and at the time when a program is running.

The loader can be called out of a program as a subroutine.

There are three instructions in the loader inserted by which the control system is activated to change the control mode from load to execute, and opening the program output file etcetera .

These three instructions must be modified with no-operation instructions before a call is made to the loader.

These modifications are: C B 7676:=C B 7677 :=C B 7700 := B 7000 ; .

The loader call is as following:

```
$ VEC( B 7600 ,# Start address of the loader #  
4      ,# Input device channel 4 #  
A BUFFER # address of the program buffer # ) ;
```

Initially BUFFER must have the value of the number of core locations which are available for storing a program. The loader modifies before returning BUFFER with the exact program length, so the program is loaded as a vector. When the buffer length is insufficient, the loader returns with an error.

The compiler output code or binary code is described on page .

The loader has a possibility to call itselfs recursively, but the compiler can not generate code for this option.

On the next pages the loader is described in Midialgol .

Note: B 4400 is a JMS Z instruction, which is an indirect subroutine call to the subroutine entry table in page 0 .

In the program example of an executive, the programmed loader call is used.

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

THE JOURNAL OF THE

The LOADER IN MIDIALGOL.

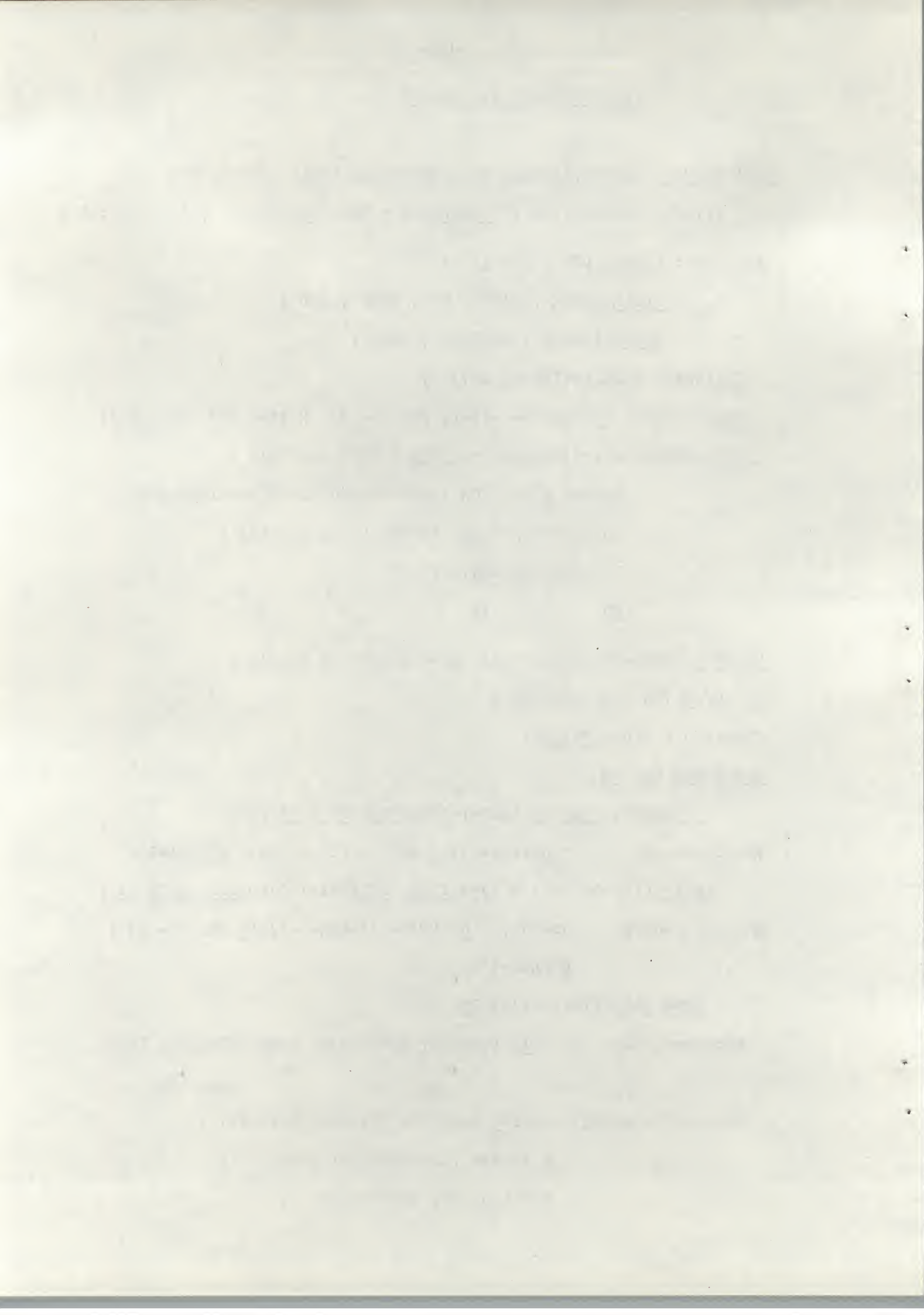
```

SUBR( FORMAL DEVICE,APVEC # program buffer begin address # ;
      PP:=PL:=BASE:=APVEC ; LABEL REC ; MAX:=-C(APVEC)-1;C(APVEC) :=0 ;

SUBR(REC: FORMAL DEV , APV ; ,
      LOCAL CHECK , SAVE , SV , TEMP , END ;
      LABEL ABSREL , LOWHIGH , READ ;
      EXPR(READ: CHECK:=(IN:= R DEV) );
      EXPR(LOWHIGH: (V READ ← -2)+(V READ ← 4) # read two frames# );
      EXPR(ABSREL: SV:=(IN+IF SV=-1 THEN 0 ELSE BASE FI) ;
            TEMP:= V LOWHIGH ; # backwards data settings #
            WHILE TEMP/=0 DO TEMP:= C (TEMP+BASE) ;
            C(TEMP+BASE):=SV
            OD
      );

WHILE V READ = 0 DO SKIP OD; # skip leading zero's #
IF IN/=B 125 THEN HALT FI ;
CHECK:=0 ; END:= FALSE ;
WHILE NOT END DO
  IF IN&1=1 THEN IF (MAX+:1)=0 THEN HALT FI ;
  #subroutine# C(PP+:1):=B 4400 + (IN ← -1); C PL+:1
  ELIF ((IN ← -1) & 1)=0 THEN IF (MAX+:1)=0 THEN HALT FI ;
  #12 bits word# H:=IN ; C (PP+:1):=(H ← -1)+(V READ ← 4) ;
  C PL+:1
  ELSE CASE (IN ← -1)+1 IN
    #program end# IF (CHECK+V LOWHIGH)=0 THEN END:=TRUE ELSE
    HALT FI ,
    #recursive call# ARG:=V LOWHIGH; SV:=-1 ; IN:=PP+1 ;
    V ABSREL ; SAVE:=BASE; BASE:=PP ;
    REC(ARG,SV); BASE:=SAVE ,

```



#absolute# SV:=-1; IN:=V LOWHIGH ; V ABSREL ,

#relative# SV:=0 ; IN:=V LOWHIGH ; V ABSREL

ESAC

FI

OD);

REC(DEVICE , APVEC));



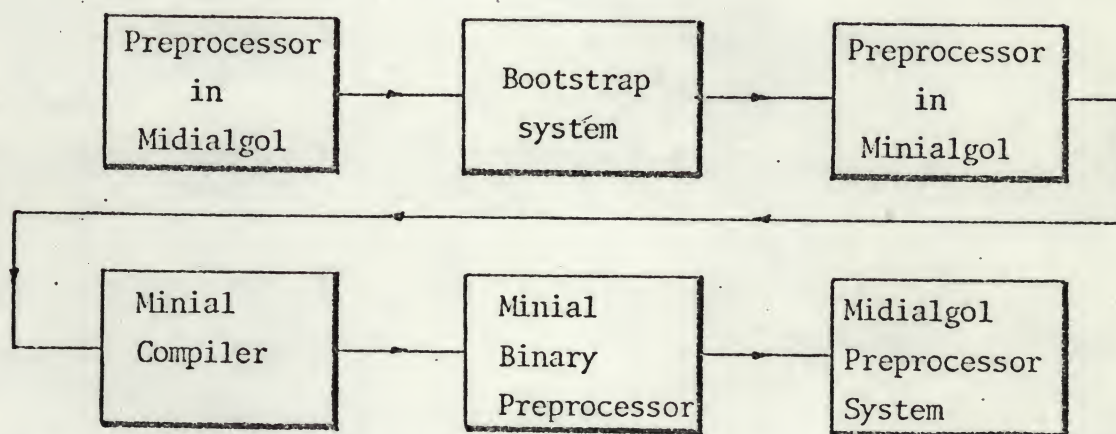
6.9 THE BOOTSTRAP SYSTEM.

The bootstrap preprocessor is created out of a minimum preliminary version of the current preprocessor written in Minial.

The bootstrap preprocessor system has more core locations free for the identifier list of its preprocessor.

With this preprocessor it is possible to translate a preprocessor written in Midial to Minial (this takes about 5 minutes).

Thereafter it can be compiled with the MIDICM compiler system , and loaded as described in the chapter about the preprocessor (page 95), and saved as a new Midial system.



On the next pages the listing of the preprocessor written in Midial and the listing of the bootstrap preprocessor are inserted.

Behind these listings, the listing of the bootstrapped Midial preprocessor is added.

The first part of the paper is devoted to a discussion of the
 various methods which have been proposed for the determination of
 the rate of reaction between a solid and a liquid. It is shown that
 the most reliable method is that of measuring the change in weight
 of the solid as the reaction proceeds. This method is applicable to
 all cases in which the solid is insoluble in the liquid and the
 reaction is not too rapid. In cases where the reaction is too rapid
 for this method to be applicable, other methods must be used.



In the second part of the paper, the author discusses the effect of
 various factors on the rate of reaction. It is shown that the rate of
 reaction is affected by the surface area of the solid, the concentration
 of the liquid, and the temperature. The effect of each of these factors
 is discussed in detail, and it is shown that the rate of reaction
 increases with increasing surface area, increasing concentration, and
 increasing temperature.

'BEGIN #MIDIALGOL TO MINIALGOL PREPROCESSOR,
WRITTEN IN MIDIALGOL

DATE 75/01/17
BY H. LUIJF
T.H. DELFT , THE NETHERLANDS.

LIST OF VARIABLES :

INPUTCHAR	-LAST INPUTTED CHARACTER
CLASS	-INPUT CHARACTER CLASS
GCNT	-G-COUNT
XCNT	-X-COUNT
LCNT	-L-COUNT
YCNT	-Y-COUNT
ELIFCNT	-ELIF-COUNT
LISTBEG	-NAME LIST BEGIN POINTER
LISTEND	-NAME LIST END POINTER
SETONLST	-CURRENT POINTER FOR SET ON LIST OPERATIONS
LOKUPLST	-NAME LIST CURRENT POINTER FOR LOOKUP OPERATIONS
SCOPE	-CURRENT LEVEL
PARLEV	-PARAMETER LEVEL
LEVCLASS	-NAME BUFFER : LEVEL AND CLASS
CHAR12	-FIRST TWO CHARACTERS OF NAME
CHAR34	-CHARACTERS 3 AND 4
CHAR56	-CHARACTERS 5 AND 6
IDENTNR	-CHAIN NUMBER OF NAME
FLAGV	-FLAG V DETECTED.
FLGHLP	-COMMON VARIABLE
FLAGCOMHA	-FLAG FOR PREVENTING THE ;) STRUCTURE: CONVERT TO ;SKIP)
FLGFOUND	-FLAG FOUND IN TABLE
HELP1	-COMMON VARIABLE
HELP3	-COMMON VARIABLE
TABLENR	-MIDIAL SPECIAL WORD CHAIN NUMBER
ELEMNR	-NUMBER OF ELEMENTS
CHAR1	-CHARACTER 1
CHAR2	-CHARACTER 2
CHAR3	-CHARACTER 3
CHAR4	-CHARACTER 4
CHAR5	-CHARACTER 5
CHAR6	-CHARACTER 6

PROGRAM DRIVER

'BEGIN #INITIALISE#

IVECTOR LEVCLASS, CHAR12, CHAR34, CHAR56, IDENTNR ;
IVECTOR ELEMNR, CHAR1, CHAR2, CHAR3, CHAR4, CHAR5, CHAR6 ;
IGLOBAL TABBEG , TABEND ;

!DCL(1A 1W0-1A 1X0-1B0060);
!P 3*="(DCL(100));"

FLGFOUND:=0; ELIFCNT:=YCNT:=SCOPE:=0; PARLEV:=1;
LISTEND:=(LISTBEG:=LOKUPLST:=SETONLST:=1AY 33
#SET AFTER X LIST#)+ 5 ;
!C LOKUPLST:=-1 ; #CREATE 1 LIST CELL #


```

IV RESETFLG ; #RESET FLAGSH
LCNT:=GCNT:=XCNT:=20 ; #PRESET X,L AND G COUNT#
IV INCHAR ; #READ AND CLASSIFY#

```

#TEXT SCREENING ON THE MAIN LEVEL#

```

!WHILE CLASS/=3 !DO
  !IF CLASS=1 !THEN IV NAME WIDENTIFIER/LABEL#
  !ELIF CLASS=4 !THEN IV MIDIALSYM #MIDIAL SYMBOL#
  !ELIF CLASS=7 !THEN IV COMMENT #COMMENT DETECTED#
  !ELIF CLASS=14 !THEN IV SPLNTAB #SP, LN, LF#
  !ELSE !IF CLASS=5 !THEN IV PUSHSCOPE
    #OPEN PARENTHESIS DETECTED#
  !ELIF CLASS=6 !THEN IV POPSCOPE
    #CLOSING PARENTHESIS DETECTED#
  !ELIF CLASS=8 !THEN FLGCOMMA:='TRUE #; DETECTED #
  !ELIF CLASS=12 !THEN IV RESETFLG; IV NEXTCHR #X DETECTED
    RESET FLGCOMMA; OUT X AND FOLLOWING CHAR.#
  !ELIF CLASS=13 !THEN IV RESETFLG; IV NEXTCHR; !WHILE CLASS/=13
    !DO IV NEXTCHR !DO #SKIP " " STRING#
  !ELSE IV RESETFLG #OUTPUT CHARACTER#
  !FI ; IV NEXTCHR
!FI
!DO ;

```

IV COMMASKIP; #TEST ;#

```

  #OUTPUT MISSING CLOSING PARENTHESIS#
!IF SCOPE>0 !THEN !P 1*="0".!DECS(SCOPE)." ) MISSING0"
  !WHILE SCOPE>0 !DO
    IV POPSCOPE ; # DELETE LEVEL#
    IV PRINTCS #PRINT#
  !DO
!FI;
!P 3*='VEC(X),X'; 'LN,'LN,'B0232) #OUTPUT THE ) OF THE (DCL....#
!END ;

```

#-----#

!EXPR(MIDIALSYM: #MIDIAL TO MINIAL - STANDARD WORDS #

#READ MIDIAL SYMBOL STRING AND SEARCH IN TABLE#

```

IV INCHAR ; #READ SYMBOL#
IV BUILDNAM ; #READ SYMBOL STRING#
HELP1:=TABBEG ; IV FLGSET #FLGHLP:='TRUE# ;
!WHILE (HELP1<=TABEND)&FLGHLP !DO

```

```

  TABLENR:=(!IF (HELP1 !HD !A ELENNR) #IN TABLE ? #
    &(!A ELENNR !HD HELP1)

```

```

    !THEN IV FLGRESET #FLGHLP:='FALSE#; !C (HELP1+!C HELP1+1)
    !ELSE HELP1+:=!C HELP1+2 ; 0
    !FI

```

!DO ;

!IF (TABLENR&'B0010)='B0010 !THEN

#FI 00 ESAC END#

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE

OFFICE OF THE DEAN OF THE FACULTY

CHICAGO, ILL.

1912

TO THE HONORABLE THE PRESIDENT OF THE UNIVERSITY
OF CHICAGO
FROM THE DEAN OF THE FACULTY

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY HAS BEEN ADVISED BY THE DEAN OF THE
FACULTY OF THE RECEIPT OF YOUR LETTER OF THE 10TH
INSTANT.

THE FACULTY HAS BEEN ADVISED BY THE DEAN OF THE
FACULTY OF THE RECEIPT OF YOUR LETTER OF THE 10TH
INSTANT.

Yours very truly,
The Dean of the Faculty

THE UNIVERSITY OF CHICAGO

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.

THE FACULTY OF THE UNIVERSITY OF CHICAGO
HAS THE HONOR TO ACKNOWLEDGE THE RECEIPT OF YOUR
LETTER OF THE 10TH INSTANT.


```

BEGIN
    #IF PREVIOUS=; THEN OUTPUT ;SKIP #
    IV POPSCOPE ; #POP LEVEL AND DELETE#
    IF TABLENR='B0012' THEN #00# IV PRINTCS
    ELSE IV PRINTWD #OUTPUT STANDARD WORD#
    IFI
END
ELSE
BEGIN
    IV RESETFLG ; #RESET FLGCOMMA#
    IF TABLENR=0 THEN #NOT FOUND# IV PRINTWD
    #OUTPUT SYMBOL#
    ELIF (TABLENR&'B7770')=0 THEN
    BEGIN
        IV PUSHSCOPE ; #PUSH LEVEL#
        IV PRINTWD ; #OUTPUT SYMBOL#
        IF TABLENR='B0002' THEN #00# IP 3:=%( IFI
    END
    ELIF TABLENR='B0401' THEN #GOTO# FLAGV:='TRUE'; IV PRINTWD
    ELIF TABLENR='B2000' THEN IV PRINTWD; CHAR34:=PARLEV;
    CHAR12:=YCNT; #SRR OR EXPR#
    ELEMR:='B0000'+(PARLEV:=SCOPE+1); IV ADDTOLST;
    YCNT:=0
    ELIF TABLENR='B0021' THEN #VECTOR LIST# IDLIST(4)
    ELIF TABLENR='B0022' THEN #ARRAY LIST# IDLIST(5)
    ELIF TABLENR='B0041' THEN #GLOBAL LIST# IDLIST(3)
    ELIF TABLENR='B0042' THEN #LABEL LIST# IDLIST(2)
    ELIF TABLENR='B1000' THEN #ELIF# ELIFCNT+=1;
    IP 3*="ELSE IF"
    ELIF TABLENR='B0100' THEN
    #FORMAL OR LOCAL LIST# IDLIST(1);
    IP 3*="DCL(" .IDCS( YCNT ) .");"
    ELSE IF TABLENR='B0201' THEN #VECH#
    IV SUBCHAR; #PUSH AND SPECIAL#
    INPUTCHAR:=%(
    ELIF TABLENR='B0400' THEN #V# FLAGV:='TRUE ;
    INPUTCHAR:=%V
    IFI ;
    IV NEXTCHR
    IFI
END
IFI
))
EXPR(PRINTWD: IP3*='A ELEMR #OUTPUT STANDARD WORD# );

```

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
DEPARTMENT OF CHEMISTRY

REPORT OF THE
COMMISSIONERS OF THE
UNIVERSITY OF CHICAGO

FOR THE YEAR
1900-1901

CHICAGO, ILL.
1901

PRINTED BY THE
UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILL.
1901

CHICAGO, ILL.
1901


```

'EXPR(INCHAR: CLASS:=( 'IF ((INPUTCHAR='R 3) >=%A)&(INPUTCHAR<=%Z)
    'THEN #ALPHABETICAL INPUT# 1
    'ELIF (INPUTCHAR >=%0)&(INPUTCHAR<=%9) 'THEN #NUMERIC# 2
    'ELIF INPUTCHAR='B0232 'THEN 3
        #CTRL Z #
    'ELIF INPUTCHAR=%' 'THEN #MIDIAL SYMBOL# 4
    'ELIF INPUTCHAR=%( 'THEN #OPENING PARENTHESIS# 5
    'ELIF INPUTCHAR=%) 'THEN #CLOSING PARENTHESIS# 6
    'ELIF INPUTCHAR=%# 'THEN #COMMENT DETECTED# 7
    'ELIF INPUTCHAR=%; 'THEN #END OF LIST OR STATEMENT# 8
    'ELIF INPUTCHAR=%: 'THEN #LABEL DECLARED# 10
    'ELIF (INPUTCHAR=%=)!(INPUTCHAR=%0) 'THEN #ASSIGN# 11
    'ELIF INPUTCHAR=%% 'THEN #CHARACTER VALUE# 12
    'ELIF INPUTCHAR=%" 'THEN # " " STRING# 13
    'ELIF (INPUTCHAR=SP)!(INPUTCHAR=0)!(INPUTCHAR='B211)!
        (INPUTCHAR='B212)!(INPUTCHAR='B200)!(INPUTCHAR='LN)
        ! (INPUTCHAR='B214)
    'THEN #SPACE,TAB,LINE FEED,CARR.RETURN,CNTRL L# 14
    'ELSE 20
    'FI )));

```

```

'EXPR(PRINT: #PRINT ONE CHARACTER# 'P 3:=INPUTCHAR);

```

```

'EXPR(NEXTCHR: #PRINT,READ,CLASSIFY# 'V PRINT;'V INCHAR);

```

```

'EXPR(PUSHSOPE: #PUSH LEVEL# LEVCLASS:='B5000+(SCOPE+:1);
    CHAR12:=ELIFCNT ; #STORE ELIFCOUNT# 'V RESETFLG ;
    'V ADDTOLST; ELIFCNT:=0 );

```

```

'EXPR(POPSOPE: #POP LEVEL#
    'V COMMASKIP ; #TEST COMMAFLAG#
    #DELETE FROM LIST# LOKUPLST:=LISTBEG;
    'WHILE ELIFCNT>0 'DO ELIFCNT-=1;'V PRINTCS 'CD;
    'WHILE LOKUPLST<LISTEND 'DO
        'IF ((HELP1:= 'C LOKUPLST)='B6000+SCOPE)!
            (HELP1='B6000+SCOPE+1) 'THEN
            'C LOKUPLST:=-1;YCNT:= 'C(LOKUPLST+1);
            PARLEV:= 'C(LOKUPLST+2) #RESTORE #
        'ELIF HELP1='B0777=SCOPE 'THEN
            HELP1-=SCOPE;
            'IF HELP1='B4000 'THEN INPUTCHAR:=%;
            'ELIF HELP1='B5000 'THEN ELIFCNT:=
                'C(LOKUPLST+1)
            'FI; 'C LOKUPLST:=-1
        'FI ;
        LOKUPLST+=5 #NEXTCHR NAME#
    'OD;
    SCOPE-=1;
    SETONLST:=LOKUPLST:=LISTBEG #RESET LIST POINTERS#
)

```

```

'EXPR(PRINTCS: 'P 3:=%) #OUTPUT A ) SYMBOL#);

```

```

'EXPR(ADDTOLST: #SET NAME ON LIST# 'V FLGSET #FLGHLP:='TRUE#;
    'WHILE FLGHLP 'DO
        'IF SETONLST<LISTEND 'THEN
            'IF 'C SETONLST=-1 'THEN 'V NAMTOLST
                'ELSE SETONLST+=5
            'FI
            'ELSE LISTEND+=5;'V NAMTOLST #COPY AFTER LIST#
            'FI
        'OD
    );

```



```

'EXPR(NAMTOLST: #COPY NAME FROM BUFFER TO LIST CELL#
  HELP3:=1A LEVCLASS-1;HELP1:=SETONLST-1;
  'WHILE (HELP3+:=1)<1A SETONLST 'DO
    'C (HELP1+:=1):='C HELP3 'DO ;
  'V FLGRESET #FLGHLP:='FALSE#);

'EXPR(COMMENT: #SKIP COMMENT# 'V NEXTCHR ;'WHILE CLASS/=7
  'DO 'V NEXTCHR 'DO ; 'V NEXTCHR );

'EXPR(BUILDNAM: #NAME DETECTED;SETUP BUFFER#
  HELP1:=LEVCLASS:=CHAR12:=CHAR34:=CHAR56:=IDENTNR:=
  ELEMNR:=CHAR1:=CHAR2:=CHAR3:=CHAR4:=CHAR5:=CHAR6:=0 ;
  'V FLGSET #FLGHLP:='TRUE# ;
  'WHILE (FLGHLP&(HELP1<6)) 'DO
    'IF (CLASS=1)!(CLASS=2) 'THEN
      'BEGIN 'C (1A CHAR1+HELP1):=INPUTCHAR ;
      INPUTCHAR-:=1B0257;#BUILD NAME#
      'C (1A CHAR12+(HELP1/2))+:=(1A (HELP1-((HELP1/2)*2))=0
      'THEN INPUTCHAR06 'ELSE INPUTCHAR 'FI);
      'V INCHAR;#READ AND CLASSIFY#
      ELEMNR:=(HELP1+:=1)
    'END
    'ELSE 'V FLGRESET #FLGHLP:='FALSE#
    'FI
  'WHILE FLGHLP&((CLASS=1)!(CLASS=2)) 'DO
    'V INCHAR #NAME LONGER THAN 6 CHAR'S# 'DO );

'EXPR(FLGSET: FLGHLP:='TRUE);

'EXPR(FLGRESET: FLGHLP:='FALSE);

'EXPR(SETUPL: #L LABEL# LEVCLASS:=PARLEV );

'EXPR(SETUPY: #Y IDENTIFIER# LEVCLASS:=PARLEV+1B1000 );

'EXPR(SUBCHAR: #SPECIAL: AT LEVEL-POP THEN ) TO )#
  'V PUSHSCOPE ;LEVCLASS:=1B4000+SCOPE;'V ADDTOLST );

'EXPR(SRCHLST: #LOOKUP IN TABLE#
  LOKUPLST:=LISTBEG-5;IDENTNR:=0;FLGFOUND:='TRUE;
  'WHILE (FLGFOUND&((LOKUPLST+:=5)<LISTEND)) 'DO
    HELP1:=-1;'V FLGSET #FLGHLP:='TRUE#;
    'WHILE (FLGHLP&((HELP1+:=1)<4)) 'DO
      'IF 'C(1A LEVCLASS+HELP1)/='C(LOKUPLST+HELP1) 'THEN
        'V FLGRESET #FLGHLP:='FALSE# 'FI
    'DO ;
    'IF FLGHLP 'THEN FLGFOUND:='FALSE ;IDENTNR:='C (LOKUPLST+4)
    'FI #NOT FOUND: LOOP#
  'DO );

'EXPR(XDEFSRCH: #SEARCH X-NAME# LEVCLASS:=1B3000;'V SRCHLST;
  'IF IDENTNR=0 'THEN #A NEW NAME#
  'V ( 'EXPR (XDEFINE: LEVCLASS:=1B3000 ; IDENTNR:=(XCNT+:=1);
    'V ADDTOLST #ADD TO LIST# )
  'FI; 2X #OUTPUT X# );

'EXPR(GSRCH: #SEARCH G#
  LEVCLASS:=1B2000;#SETUP G# 'V SRCHLST #AND SEARCH# );

'EXPR(ALLSCOPES: #SEARCH LOOP UNTIL LEVEL 0#
  LEVCLASS+:=1; IDENTNR:=0 ;
  'WHILE (((LEVCLASS-:=1)&1B0777)/=0) & (IDENTNR=0) 'DO

```

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

OFFICE OF THE DEAN
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

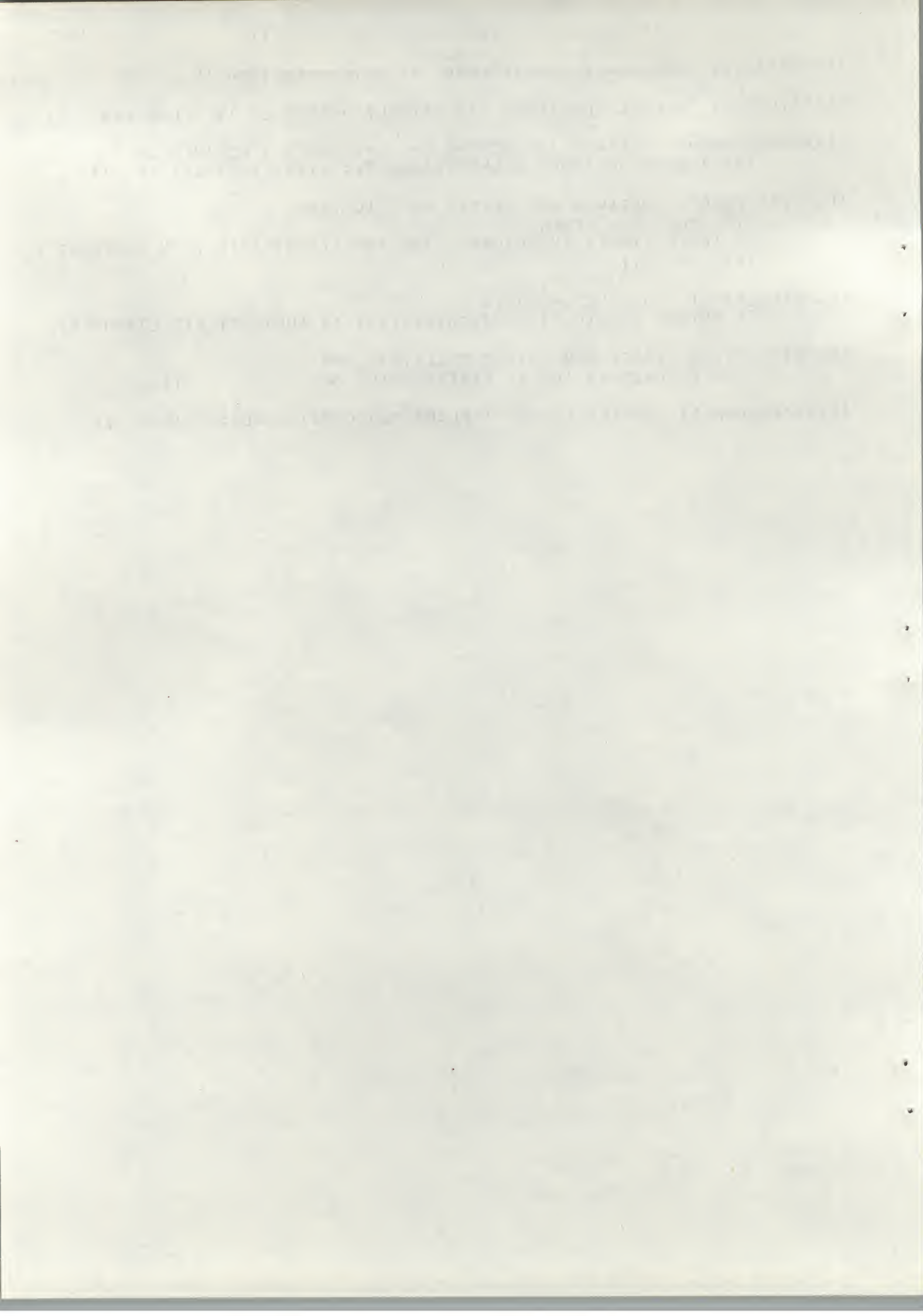
THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF THE HISTORY OF ARTS
AND ARCHITECTURE
1100 EAST 58TH STREET
CHICAGO, ILLINOIS 60637


```
'EXPR(YSRCH: #SEARCH Y IDENTIFIER# IV SETUPY;#SETUP# IV ALLSCOPES #SEAR
'EXPR(LSRCH: #SEARCH L-LABEL# IV SETUPL; #SETUP L# IV ALLSCOPES );
'EXPR(GDEFSRCH: #SEARCH AND DEFINE G# IV GSRCH ; #SEARCH G#
    'IF IDENTNR=0 'THEN IDENTNR:=(GCNT+:1); IV ADDTOLST 'FI );
'EXPR(YDEFSRCH: #SEARCH AND DEFINE Y# IV YSRCH ;
    'IF IDENTNR=0 'THEN
    IV 'EXPR (YDEF: IV SETUPY ; IDENTNR:=(YCNT+:1) ; IV ADDTOLST )
    'FI );
'EXPR(LDEFINE: #DEFINE L-LABEL#
    IV SETUPL ; IDENTNR:=(LCNT+:1) ; IV ADDTOLST #TO LIST# );
'EXPR(SPLNTAB: #SKIP AND OUTPUT CR,LF,TAB,SP#
    'WHILE CLASS=14 'DO IV NEXTCHR 100 );
'EXPR(RESETFLG: #RESET FLGCOMMA FLAG# FLGCOMMA:=FLAGV:='FALSE );
```




```

'EXPR(IDNROUT: #OUTPUT IDENTNR#   'P 3*=('DECS(IDENTNR))." "   ))
'EXPR(COMMASKIP: #TEST  )) COMBINATION;OUTPUT "SKIP" IF SO#
    'IF FLGCOMMA 'THEN 'P 3*="SKIP ";IV RESETFLG #RESET#   IF

'EXPR(NAME: #IDENTIFIER DETECTED#   IV BUILDNAM;#READ NAME#
    'IF CLASS=7 #COMMENTS# 'THEN IV COMMENT 'FI;
    IV SPLNTAB ;   #SKIP SPACES,LINE FEEDS ETC#
    'IF CLASS=10 'THEN   #: DETECTED#
    'BEGIN
        IV INCHAR; #READ NEXTCHR#
        'P 3:= 'IF CLASS=11   #:=# 'THEN
            'BEGIN
                IV YSRCH ; #SEARCH Y#
                'IF IDENTNR=0 'THEN IV XDEFSRCH #X IDENTIFIER#
                    'ELSE   XY#Y IDENTIFIER#
                'FI
            'END
            'ELSE   #LABEL#
                IV LSRCH; #SEARCH L#
                'IF IDENTNR=0 'THEN #G#   IV GDEFSRCH; %G
                    'ELSE   %L
                'FI
            'FI ;
            IV IDNROUT; #OUTPUT CHAIN NUMBER# 'P 3:=%;
        'END
    'ELSE
        'IF CLASS=5   #%(#   'THEN   #PROCEDURE NAME#
            'BEGIN
                IV GDEFSRCH; #SEARCH AND DEFINE G#
                IV SUBCHAR; #SUB CHAR. TO LIST#
                'P 3*="S[G";IV IDNROUT;INPUTCHAR:=%,;IV NEXTCHR
                    #OUTPUT $[G.....#
            'END
        'ELSE
            'P3:= 'BEGIN
                IV LSRCH; #LOCAL LABEL?#
                'IF IDENTNR=0 'THEN
                    'IF FLAGV 'THEN IV GDEFSRCH;   %G   #FUNCTION#
                    'ELSE
                        'BEGIN   IV YSRCH ; #SEARCH Y#
                            'IF IDENTNR=0 'THEN
                                'BEGIN IV GSRCH ;#SEARCH G=LABEL#
                                    'IF IDENTNR=0 'THEN   IV XDEFSRCH
                                        'ELSE   %G
                                    'FI
                                'END
                            'ELSE   %Y
                            'FI
                        'END
                    'FI
                'ELSE   %L
                'FI
            'END ;
            IV IDNROUT   #PRINT NUMBER#
        'FI
    'FI
    ;IV RESETFLG   #RESET FLGCOMMA AND FLAGV#   ))

'SUBR(IDLIST: #LIST OF NAMES#   'FORMAL SWITCH ;   TABLENR:=1TRUE;
    'WHILE TABLENR 'DO

```

THE UNIVERSITY OF CHICAGO PRESS

1911

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

THE UNIVERSITY OF CHICAGO PRESS

IF CLASS=8 THEN TABLENR:=FALSE #; IS LIST END#

ELIF CLASS=1 THEN

BEGIN

IV BUILDNAM ; #READ NAME#

CASE SWITCH IN

#Y# IV YDEF ,

#L# IV LDEFINE ,

#G# IV GDEFSRCH ,

#X# IV XDEFINE , #VECTOR LIST#

#X# BEGIN IV XDEFINE; HELP1:=0 ;HARRAY#

WHILE CLASS/=2 DO IV INCHAR 'OD;

#READ UNTIL A NUMERICAL CHARACTER#

WHILE CLASS=2 DO

HELP1:=(HELP1*10)+(INPUTCHAR-%0);

IV INCHAR

OD ; XCNT+=HELP1

END

ESAC

END

ELSE IV INCHAR

IF

OD ; IV INCHAR

)

TABLE(TABSEG: 2,%I,%F,'B0001,
2,%O,%O,'B0002,
4,%C,%A,%S,%E,'B0003,
5,%B,%E,%G,%I,%N,'B0004,

2,%F,%I,'B0011,
2,%O,%O,'B0012,
4,%E,%S,%A,%C,'B0013,
3,%E,%N,%O,'B0014,

6,%V,%E,%C,%T,%O,%R,'B0021,
5,%A,%R,%R,%A,%Y,'B0022,

6,%G,%L,%O,%B,%A,%L,'B0041,
5,%L,%A,%B,%E,%L,'B0042,

6,%F,%O,%R,%M,%A,%L,'B0100,
5,%L,%O,%C,%A,%L,'B0100,

3,%V,%E,%C,'B0201,

1,%V,'B0400,
4,%E,%L,%I,%F,'B1000 ,

4,%S,%U,%R,%R,'B2000,

TABEND: 4,%E,%X,%P,%R,'B2000)

END;

BEGIN #MIDIAL TO MINIAL PREPROCESSOR, VERSION BOOTSTRAP
 DATE 74/12/16
 BY P.LUIJF
 T.H. DELFT

LIST OF VARIABLES :

X1 -COMMON VARIABLE
 X2 -LAST INPUT CHARACTER
 X3 -INPUT CHARACTER CLASS
 X4 -G-COUNT
 X5 -X-COUNT
 X7 -Y-COUNT
 X8 -NAME LIST BEGIN POINTER
 X9 -NAME LIST END POINTER
 X10 -NAME LIST CURRENT POINTER FOR LOOKUP OPERATIONS
 X11 -CURRENT LEVEL
 X12 -NAME BUFFER : LEVEL AND CLASS
 X13 -FIRST TWO CHARACTERS OF NAME
 X14 -CHARACTERS 3 AND 4
 X15 -CHARACTERS 5 AND 6
 X16 -CHAIN NUMBER OF NAME
 X17 -CURRENT POINTER FOR SET ON LIST OPERATIONS
 X18 -COMMON VARIABLE
 X19 -FLAG FOR PREVENTING THE ;) STRUCTURE: CONVERT TO ;SKIP)
 X20 -FLAG FOUND IN TABLE
 X21 -COMMON VARIABLE
 X22 -MIDIAL SPECIAL WORD CHAIN NUMBER
 X23 -NUMBER OF ELEMENTS
 X24 -CHARACTER 1
 X25 -CHARACTER 2
 X26 -CHARACTER 3
 X27 -CHARACTER 4
 X28 -CHARACTER 5
 X29 -CHARACTER 6
 X30 -FLAG V DETECTED.
 X31 -ELIF COUNTER

PROGRAM DRIVER

BEGIN #INITIALISE#

X7:=X11:=X31:=0;
 X8:=X10:=X17:=AX32 ; #SET AFTER X-LIST#
 X20:=FALSE ; VG34 ; #RESET FLAGS#

#-----#

X4:=X5:=0 ; #PRESET X AND G COUNT#

#FOR THIS BOOTSTAP VERSION NOT 20 BUT 0 #

#-----#

X9:=X10+5 ; CX10:=-1 ; #CREATE 1 CELL#
 VG3 ; #READ AND CLASSIFY#
 DCL(AW0-AX0+80060);
 P3:=%(;

THE UNIVERSITY OF CHICAGO
LIBRARY
1207 EAST 58TH STREET
CHICAGO, ILL. 60637
U.S.A.
TEL. 773-936-5000
FAX 773-936-5000
WWW.CHICAGO.EDU
CHICAGO, ILL. 60637
U.S.A.
TEL. 773-936-5000
FAX 773-936-5000
WWW.CHICAGO.EDU


```

WHILE X3/=3 DO
  BEGIN
    IF X3=12 THEN (VG34;VG5;VG5) #X DETECTED
      RESET X19;OUT % AND FOLLOWING CHAR# ELSE
    IF X3=1 THEN VG40 #IDENTIFIER/LABEL# ELSE
    IF X3=4 THEN VG1 #MIDIAL SYMBOL# ELSE
    IF X3=5 THEN BEGIN VG34;VG6;VG5 END #OPEN PARENTHESIS# ELSE
    IF X3=6 THEN (VG35;VG7;VG5) #CLOSING PARENTHESIS DETECTED,
      TEST PREVIOUS CHAR=; ,POP ,READ # ELSE
    IF X3=7 THEN VG11 #COMMENT DETECTED# ELSE
    IF X3=8 THEN (X10:=TRUE;VG5) #; DETECTED# ELSE
    IF X3=13 THEN ( VG34;VG5; WHILE X3/=13 DO VG5; VG5)
      #SKIP " " STRING# ELSE
    IF X3=14 THEN VG31 #SP,LN,LF# ELSE
      (VG34;VG5) #OUTPUT CHARACTER# FI FI FI FI FI FI FI FI FI
  END ;
  VG35; #TEST ;#

```

```

      #OUTPUT MISSING CLOSING PARENTHESIS#
  IF X11>0 THEN P1:=%W ;
    WHILE X11>0 DO
      BEGIN X2:=%); VG7 ;#POP AND DELETE LEVEL#
        VG4 #PRINT#
      END
    FI;

```

```

  P3*=(%), %), LN , LN , B0232) #OUTPUT THE ) OF THE (DCL....#
END ;

```

```

#-----#

```

```

EXPR(G1: #MIDIAL TO MINIAL STANDARD WORDS CONVERTING#

```

```

  #READ MIDIAL SYMBOL STRING AND SEARCH IN TABLE#
  X23:=X24:=X25:=X26:=X27:=X28:=X29:=0 ;
  VG3 ; #READ SYMBOL#
  VG14 ; #READ SYMBOL STRING#
  X1:=G50 ; X18:=TRUE ;
  WHILE ((X1<=G51)&X18) DO
    BEGIN
      X22:=(IF (X1 HD AX23) THEN #IN TABLE#
        BEGIN X18:=FALSE ; C(X1+CX1+1) END
        ELSE BEGIN X1+:=CX1+2 ; #
          FI
      END ;

```

```

  IF ((X22&B0010)=B0010) THEN
    BEGIN #FI OD ESAC END#
      VG35 ; #IF PREVIOUS=; THEN OUT ;SKIP #
      VG7 ; #POP LEVEL AND DELETE#
      IF X22=B0012 THEN #OD# P3:=%)
        ELSE P3*=AX23 #OUTPUT MINIAL SYMBOL#
      FI
    END
  ELSE
    BEGIN
      VG34 ; #RESET X19#

```

Vol. 14, No. 19
May 1, 1914

Subscription price, Five Dollars Per Annum in Advance

Single Copies, Fifteen Cents

Entered as Second-Class Matter, May 2, 1912

Postage Paid at Chicago, Ill.

Acceptance for mailing at special rate of postage provided for in Act of October 3, 1917

Authorizes sale at wholesale price of ten cents per copy

Copyright, 1914, by American Medical Association

Printed at the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Published by the American Medical Association

Subscription orders, notices, and correspondence should be sent to the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Entered as Second-Class Matter, May 2, 1912

Postage Paid at Chicago, Ill.

Acceptance for mailing at special rate of postage provided for in Act of October 3, 1917

Authorizes sale at wholesale price of ten cents per copy

Copyright, 1914, by American Medical Association

Printed at the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Published by the American Medical Association

Subscription orders, notices, and correspondence should be sent to the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Entered as Second-Class Matter, May 2, 1912

Postage Paid at Chicago, Ill.

Acceptance for mailing at special rate of postage provided for in Act of October 3, 1917

Authorizes sale at wholesale price of ten cents per copy

Copyright, 1914, by American Medical Association

Printed at the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Published by the American Medical Association

Subscription orders, notices, and correspondence should be sent to the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Entered as Second-Class Matter, May 2, 1912

Postage Paid at Chicago, Ill.

Acceptance for mailing at special rate of postage provided for in Act of October 3, 1917

Authorizes sale at wholesale price of ten cents per copy

Copyright, 1914, by American Medical Association

Printed at the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Published by the American Medical Association

Subscription orders, notices, and correspondence should be sent to the American Medical Association, 535 North Dearborn Street, Chicago, Ill.

Entered as Second-Class Matter, May 2, 1912


```

IF X22=0 THEN #NOT FOUND# P3*:=AX23 #OUTPUT SYMBOL#
ELSE
  IF ((X22&B7770)=0) THEN
    BEGIN
      VG6 : #PUSH LEVEL#
      P3*:=AX23; #OUTPUT SYMBOL#
      IF X22=B0002 THEN #DO# P3:=X( FI
    END
  ELSE
    IF X22=B0021 THEN #VECTOR LIST# $[G42,4)
  ELSE
    IF X22=B0041 THEN #GLOBAL LIST# $[G42,3)
  ELSE
    IF X22=B1000 THEN X31+;:=1;P3*:= "ELSE IF"
  ELSE
    IF (X22=B0101) THEN
      BEGIN
        #FORMAL LIST# $[G42,1) ;
        VG39 #OUTPUT DECLAREN#
      END
    ELSE
      IF X22=B0201 THEN #VECH#
      BEGIN
        VG20; #PUSH AND SPECIAL#
        X2:=X(
      END
    ELSE
      IF X22=B0403 THEN #IVH (X30:=TRUE;X2:=XV)
      FI
      FI
      VG5
      FI
      FI
      FI
      FI
      FI
      FI
      END
      FI

```

))

```

EXPR(G3: #READ AND CLASSIFY INPUT CHARACTER#
X3:= ( IF ((X2:=R3) >=XA)&(X2<=ZZ)) THEN #ALPHABETIC# 1 ELSE
        IF ((X2 >=X0)&(X2<=X9)) THEN #NUMERIC# 2 ELSE
        IF (X2=B0232) THEN #CNTRL Z# 3 ELSE
        IF X2=X' THEN #MIDIAL SYMBOL# 4 ELSE
        IF X2=X( THEN #OPENING PARENTHESIS# 5 ELSE
        IF X2=X) THEN #CLOSING PARENTHESIS# 6 ELSE
        IF X2=X# THEN #COMMENT DETECTED# 7 ELSE
        IF X2=X; THEN #END OF LIST OR STATEMENT# 8 ELSE
        IF X2=X: THEN #LABEL DECLARED# 10 ELSE
        IF ((X2=X=)! (X2=X0)) THEN #ASSIGN# 11 ELSE
        IF X2=XX THEN #CHARACTER VALUE# 12 ELSE
        IF X2=X" THEN # " " STRING# 13 ELSE
        IF (X2=SP)! (X2=0)! (X2=B211)! (X2=B212)! (X2=LN)! (X2=B200)
          ! (X2=B214)
        THEN #SPACE,TAB,LINE FEED,CARR.RETURN# 14
        ELSE 20 FI FI FI FI FI FI FI FI FI FI FI FI FI FI FI))

```

EXPR(G4: #PRINT ONE CHARACTER# P3:=X2);

EXPR(G5: #PRINT,READ,CLASSIFY# VG4;VG3);

THE UNIVERSITY OF CHICAGO
LIBRARY

1000
1000
1000
1000
1000

1000
1000
1000
1000
1000

1000
1000
1000
1000
1000

1000
1000
1000
1000
1000

1000
1000
1000
1000
1000

1000
1000
1000
1000
1000

1000
1000
1000
1000
1000

1000
1000
1000
1000
1000


```
EXPR(G6: #PUSH LEVEL# X12:=85000+(X11+:=1) ; X14:=X7 ; X15:=X31 ;  
#STORE Y AND ELIF COUNT# VG9; X7:=X31:=0 );
```




```

EXPR(G7: #POP LEVEL#
#DELETE FROM LIST# X10:=X8;
WHILE X31>0 DO ( X31-:=1;P3:=X) #ELIF CLOSE# );
WHILE X10<X9 DO
BEGIN IF (CX10&80777)=X11 #SAME LEVEL#
THEN IF CX10=(B4000+X11) #SPECIAL) TO J#
THEN X21:=X; FI;
IF CX10=(B5000+X11) THEN
X7:=C(X10+2); #RESTORE#
X31:=C(X10+3)
FI ;
CX10:=-1 #DELETION SYMBOL#
FI;
X10+:=5 #NEXT NAME#
END;
X11-:=1;
X17:=X10:=X8 #RESET LIST POINTERS# );

EXPR(G9: #SET NAME ON LIST# X18:=TRUE;
WHILE X18 DO
BEGIN IF X17<X9 THEN
IF CX17=-1 THEN VG10 #COPY IN LIST#
ELSE X17+:=5
FI
ELSE X9+:=5;VG10 #COPY AFTER LIST#
FI
END );

EXPR(G10: #COPY NAME FROM BUFFER TO LIST CELL#
X21:=AX12-1;X1:=X17-1;
WHILE (X21+:=1)<AX17 DO
C(X1+:=1):=CX21 ;
X18:=FALSE);

EXPR(G11: #SKIP COMMENT# VG5;WHILE X3/=7 DO VG5;VG5);

EXPR(G14: #NAME DETECTED;SETUP BUFFER#
X1:=0;X18:=TRUE;X12:=X13:=X14:=X15:=X16:=0;
WHILE (X18&(X1<6)) DO
BEGIN IF ((X3=1)!(X3=2)) THEN
BEGIN C(AX24+X1):=X2 ;
X2-:=B0257;#BUILD NAME#
C(AX13+(X1/2))+:=(IF (X1-((X1/2)*2))=0 THEN
X206 ELSE X2 FI);
VG3;#READ AND CLASSIFY#X1+:=1
END
ELSE X18:=FALSE
FI
END;
X23:=X1 ; #TRANSFER NAME LENGTH#
WHILE X18&((X3=1)!(X3=2)) DO
BEGIN VG3 #NAME LONGER THAN 6 CHAR.#
END
);

EXPR(G19: #Y IDENTIFIER# X12:=X11+B1000 #LEVEL X11#);

EXPR(G20: #SPECIAL: AT LEVEL=POP THEN ) TO J#
VG6;#PUSH LEVEL# X12:=B4000+X11;VG9#TO LIST# );

EXPR(G21: #LOOKUP IN TABLE#
X10:=X8-5;X16:=0;X20:=TRUE;

```

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
DEPARTMENT OF CHEMISTRY
530 SOUTH EAST ASIAN AVENUE
CHICAGO, ILLINOIS 60607

TO: THE DIRECTOR, NATIONAL BUREAU OF STANDARDS
WASHINGTON, D.C. 20535
FROM: DR. J. H. DUNN, JR.
DEPARTMENT OF CHEMISTRY
UNIVERSITY OF CHICAGO
CHICAGO, ILLINOIS 60607
SUBJECT: A NEW METHOD FOR THE DETERMINATION OF
COPPER IN URINE BY CATHODOLUMINESCENCE
ANALYSIS OF URINE EXTRACTS

Enclosed for the Bureau are two copies of a paper
entitled "A New Method for the Determination of
Copper in Urine by Cathodoluminescence Analysis of
Urine Extracts" by J. H. Dunn, Jr. and J. E.
Harris, published in the JOURNAL OF CLINICAL
CHEMISTRY AND MEDICAL INVESTIGATION, Vol. 12,
No. 1, 1964, pp. 1-10. The paper describes a
new method for the determination of copper in urine
by cathodoluminescence analysis of urine extracts.
The method involves the extraction of copper from
urine with a suitable organic solvent, followed by
the determination of the copper in the extract by
cathodoluminescence analysis.

The method is simple, rapid, and accurate, and
is suitable for the determination of copper in
urine in the range of 0.1 to 1.0 micrograms per
milliliter. The method has been applied to the
determination of copper in urine from patients with
Wilson's disease, and the results have been compared
with those obtained by the standard method of
determination of copper in urine by atomic absorption
spectrophotometry.


```

    WHILE ((X2=X((X1+1-5)<X0))<X0) DO
    BEGIN X1:=-1; X18:=TRUE;
    WHILE ((X18&((X1+1)<4)) DO
    IF C(AX12+X1)/=C(X10+X1) THEN X18:=FALSE FI;
    IF X18 THEN X20:=FALSE #FOUND#; X16:=C(X10+4)
    FI #NOT FOUND: LOOP#
    END
    ));

EXPR(G22: #SEARCH X-NAME# X12:=B3000; VG21; #SEARCH X#
IF X16=0 THEN #A NEW NAME#
V( EXPR (G221: X12:=B3000 ; X16:=(X5+:=1); #CHAIN#
VG9 #ADD TO LIST# ))
FI; P3:=%X #OUTPUT X# ));

EXPR(G23: #SEARCH G#
X12:=B2000; #SETUP G# VG21 #AND SEARCH# ));

EXPR(G24: #SEARCH LOOP UNTIL LEVEL 0#
X12+:=1; X16:=0;
WHILE (((X12-:=1)&B0777)/=0) & (X16=0)) DO
VG21 #SEARCH# ));

EXPR(G25: #SEARCH Y IDENTIFIER# VG19; #SETUP# VG24 #SEARCH#);

EXPR(G27: #OUTPUT X16# P3:=(DECS(X16))." " ));

EXPR(G28: #SEARCH AND DEFINE G# VG23 ; #SEARCH G#
IF X16=0 THEN X16:=(X4+:=1); VG9 FI );

EXPR(G29: #SEARCH AND DEFINE Y# VG25 ;
IF X16=0 THEN
V EXPR (G291: VG19 ; X16:=(X7+:=1) ; VG9 )
FI
));

EXPR(G31: #SKIP AND OUTPUT CR,LF,TAB,SP#
WHILE X3=14 DO VG5
));

EXPR(G34: #RESET X19 FLAG# X19:=X30:=FALSE ));

EXPR(G35: #TEST ; ) COMBINATION; OUTPUT "SKIP" IF SO#
IF X19 THEN P3:="SKIP "; VG34#RESET# FI );

EXPR(G39: #OUTPUT DECLARE# P3:="DCL( ".DECS(X7).") ;" ));

EXPR(G40: #IDENTIFIER DETECTED# VG14; #READ NAME#
IF X3=7 #COMMENT# THEN VG11 FI;
VG31 ; #SKIP SPACES, LINE FEEDS ETC#
IF X3=10 THEN #: DETECTED#
BEGIN
VG31 #READ NEXT#
IF X3=11 #:=# THEN
BEGIN
VG25 ; #SEARCH Y#
IF X16=0 THEN VG22 #X IDENTIFIER#
ELSE P3:=XY#Y IDENTIFIER#
FI;
VG27 ; #OUTPUT X16#
P3:=%: #PRINT := , READ CHAR#
END
ELSE #LABEL#
#G# P3:=%G; VG28 ; #SEARCH AND DEFINE#

```



```

FI ;
VG5      #READ NEXT#

END
ELSE BEGIN
    IF X3=5    #X(  THEN  #PROCEDURE NAME#
        BEGIN
            VG28; #SEARCH AND DEFINE G#
            VG20; #SPECIAL TO LIST#
            P3*="#$IG";VG27;X2:=X,;VG5
            #OUTPUT $IG.....#
        END
    ELSE
        BEGIN
            IF X30 THEN VG28 / P3:=XG    #FUNCTION#
        ELSE
            BEGIN    VG25 ; #SEARCH Y#
                IF X16=0 THEN
                    BEGIN VG23 ; #SEARCH G#
                        IF X16=0 THEN VG22 #SEARCH AND DEF X#
                        ELSE    P3:=XG
                    FI
                END
            ELSE    P3:=XY
            FI
        END
    END /
    VG27      #PRINT NUMBER#
FI
END
FI      ;VG34    #RESET X19 AND X30#    ))

SUBR(G42:    #LIST OF NAMES#    X22:=TRUE;
WHILE X22 DO
BEGIN
    VG31; #SKIP LF,CR,TAB,SP#
    IF X3=7 THEN VG11 FI; #SKIP COMMENT#
    IF X3=8 THEN X22:=FALSE    #; IS LIST END#
    ELSE IF X3=1 THEN
        BEGIN
            VG14 ; #READ NAME#
            CASE Y1 IN
                #Y#    VG29; ,
                #L#    SKIP ,
                #G#    VG28 ,
                #X#    VG22;
            ESAC
        END
    ELSE VG3
    FI
FI
END ;    VG3
))

```


TABLE(G50:

-121-

2,%I,%F,B0001,
2,%D,%O,B0002,
4,%C,%A,%S,%E,B0003,
5,%B,%E,%G,%I,%N,B0004,

2,%F,%I,B0011,
2,%O,%D,B0012,
4,%E,%S,%A,%C,B0013,
3,%E,%N,%D,B0014,

6,%V,%E,%C,%T,%O,%R,B0021,

6,%G,%L,%O,%B,%A,%L,B0041,

6,%F,%O,%R,%M,%A,%L,B0101,

3,%V,%E,%C,B0201,

1,%V,B0403,
4,%E,%L,%I,%F,B1000

G51:

END;

THE UNIVERSITY OF CHICAGO
LIBRARY

100 EAST 57TH STREET
CHICAGO, ILL. 60637
TEL. 371-4100
FAX 371-4100
WWW.CHICAGO.EDU
LIBRARY@CHICAGO.EDU

BEGIN #MIDIALGOL TO MINIALGOL PREPROCESSOR,
WRITTEN IN MIDIALGOL

DATE 75/01/17
BY H.LUIIJF
T.H. DELFT , , THE NETHERLANDS.

LIST OF VARIABLES :

INPUTCHAR	-LAST INPUTTED CHARACTER
CLASS	-INPUT CHARACTER CLASS
GCNT	-G-COUNT
XCNT	-X-COUNT
LCNT	-L-COUNT
YCNT	-Y-COUNT
ELIFCNT	-ELIF-COUNT
LISTBEG	-NAME LIST BEGIN POINTER
LISTEND	-NAME LIST END POINTER
SETONLST	-CURRENT POINTER FOR SET ON LIST OPERATIONS
LOKUPLST	-NAME LIST CURRENT POINTER FOR LOOKUP OPERATIONS
SCOPE	-CURRENT LEVEL
PARLEV	-PARAMETER LEVEL
LEVCLASS	-NAME BUFFER : LEVEL AND CLASS
CHAR12	-FIRST TWO CHARACTERS OF NAME
CHAR34	-CHARACTERS 3 AND 4
CHAR56	-CHARACTERS 5 AND 6
IDENTNR	-CHAIN NUMBER OF NAME
FLAGV	-FLAG V DETECTED.
FLGHLP	-COMMON VARIABLE
FLAGCOMPA	-FLAG FOR PREVENTING THE ;) STRUCTURE:CONVERT TO ;SKIP)
FLGFOUND	-FLAG FOUND IN TABLE
HELP1	-COMMON VARIABLE
HELP3	-COMMON VARIABLE
TABLENR	-MIDIAL SPECIAL WORD CHAIN NUMBER
ELEMNR	-NUMBER OF ELEMENTS
CHAR1	-CHARACTER 1
CHAR2	-CHARACTER 2
CHAR3	-CHARACTER 3
CHAR4	-CHARACTER 4
CHAR5	-CHARACTER 5
CHAR6	-CHARACTER 6

PROGRAM DRIVER

BEGIN #INITIALISE#

DCL(A W0-A X0-B00050);
P 3*="(DCL(100);"

X13 :=X14 :=X15 :=X16 :=0;X17 :=1;
X18 :=(X19 :=X20 :=Y21 :=AX 33

C X20 :=-1 ; #CREATE 1 LIST CELL #

#SET AFTER X LIST#)+ 5 ;


```
V G3 ; #RESET FLAGSH
X22 :=X23 :=X24 :=20 ; #PRESET X,L AND G COUNT#
VG4 ; #READ AND CLASSIFY#
```

#TEXT SCREENING ON THE MAIN LEVEL#

```
WHILE X25 /=3 DO(
```

```
IF X25 =1 THEN V G5 #IDENTIFIER/LABEL#
ELSE IF X25 =4 THEN V G6 #MIDIAL SYMBOL#
ELSE IF X25 =7 THEN V G7 #COMMENT DETECTED#
ELSE IF X25 =14 THEN V G8 #SP, LN, LF#
ELSE IF X25 =5 THEN V
    G9 #OPEN PARENTHESIS DETECTED#
ELSE IF X25 =6 THEN V
    G10 #CLOSING PARENTHESIS DETECTED#
ELSE IF X25 =8 THEN X26 :=TRUE #; DETECTED #
ELSE IF X25 =12 THEN VG3 ; V G11 #% DETECTED
    RESET FLGCOMMA; CUT % AND FOLLOWING CHAR.#
ELSE IF X25 =13 THEN VG3 ; VG11 ; WHILE X25 /=13
    DO( V G11 ) #SKIP " " STRING#
ELSE V G3 #OUTPUT CHARACTER#
))))FI ; V
G11 ))))FI
) ;
```

```
VG12 ; #TEST ;#
```

```
#OUTPUT MISSING CLOSING PARENTHESIS#
IF X16 >0 THEN P 1+="#".DECS(X16 )." ) MISSING0" ;
WHILE X16 >0 DO(
    V G10 ; # DELETE LEVEL#
    V G13 #PRINT#
)
```

```
FI;
```

```
P 3+=[X),X; ,LN,LN,B22321 #OUTPUT THE ) OF THE (DCL....#
```

```
END ;
```

```
#-----#
```

```
EXPR(G6 ; #MIDIAL TO MINIAL - STANDARD WORDS #
```

#READ MIDIAL SYMBOL STRING AND SEARCH IN TABLE#

```
V G4 ; #READ SYMBOL#
V G14 ; #READ SYMBOL STRING#
X27 := G1 ; V G15 #FLGHLF:=TRUE# ;
WHILE (X27 <=G2 )& X28 DO(
    X29 :=(IF ( X27 HD A X6 ) #IN TABLE ? #
        &(A X6 HD X27 )
    THEN V G16 #FLGHLF:=FALSE#; C (X27 +C X27 +1)
    ELSE X27 +=C X27 +2 ; 0
    FI
) ;
```

```
IF (X29 &B0010)=B2010 THEN
```

```
#FI OD ESAC END#
```

THE UNIVERSITY OF CHICAGO
LIBRARY

1911

THE UNIVERSITY OF CHICAGO
LIBRARY
1911

THE UNIVERSITY OF CHICAGO
LIBRARY
1911

THE UNIVERSITY OF CHICAGO
LIBRARY
1911

THE UNIVERSITY OF CHICAGO
LIBRARY
1911

THE UNIVERSITY OF CHICAGO
LIBRARY
1911

THE UNIVERSITY OF CHICAGO
LIBRARY
1911


```

      .#IF PREVIOUS# THEN OUTPUT ;SKIP #
V G10 ; #POP LEVEL AND DELETE#
IF X29 =B0012 THEN #DO# V
      G13 ELSE V G17 #OUTPUT STANDARD WORD#
FI

```

END

ELSE

BEGIN

V G3 ; #RESET FLGCOMP#

IF X29 =0 THEN #NOT FOUND# V

G17 #OUTPUT SYMBOL#

ELSE IF (X29 &B7770)=0 THEN

BEGIN

V G9 ; #PUSH LEVEL#

V G17 ; #OUTPUT SYMBOL#

IF X29 =B0002 THEN #DO# P 3:=X(FI

END

ELSE IF X29 =B0401 THEN #GOTO# X30 :=TRUE;V

G17 ELSE IF X29 =B2000 THEN VG17 ;X3 :=X17 ;

X2 :=X15 ; #SUBR OR EXPR#

X6 :=B6000+(X17 :=X16 +1);VG18 ;

X15 :=0

ELSE IF X29 =B0021 THEN #VECTOR LIST# \$(G19 ,4)

ELSE IF X29 =B0022 THEN # ARRAY LIST# \$(G19 ,5)

ELSE IF X29 =B0041 THEN #GLOBAL LIST# \$(G19 ,3)

ELSE IF X29 =B0042 THEN #LABEL LIST# \$(G19 ,2)

ELSE IF X29 =B1000 THEN #ELIF# X14 +=1;

P 3*="ELSE IF"

ELSE IF X29 =B0100 THEN

#FORMAL OR LOCAL LIST# \$(G19 ,1);

P 3*="DCL(" ,DECS(X15) ,")";

ELSE IF X29 =B0201 THEN #VECH

VG20 ; #PUSH AND SPECIAL#

X31 :=X1

ELSE IF X29 =B0400 THEN #V# X30 :=TRUE ;

X31 :=XV

)FI ;

V

G11))))))))FI

END

FI

);

EXPR(G17 ; P3*:=A X6 #OUTPUT STANDARD WORD#);

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
DEPARTMENT OF CHEMISTRY
CHICAGO, ILLINOIS 60637

TO THE EDITOR:
I am writing to you to inform you of the results of my research on the properties of the new material which I have discovered. This material has many unique properties and I believe it will be of great value to the scientific community. I have attached a copy of my report for your information. I would be pleased to discuss the results of my research with you at any time. I am sure that you will find this information of interest. I am sure that you will find this information of interest. I am sure that you will find this information of interest.

Very truly yours,
[Signature]
[Name]
[Address]
[City, State, Zip]


```

EXPR(G4 : X25 :=(IF ((X31 :=R 3) >=%A)&(X31 <=%Z)
THEN      #ALPHABETICAL INPUT# 1
ELSE IF ( X31 >=%0)&(X31 <=%9) THEN #NUMERIC# 2
ELSE IF X31 =B0232 THEN 3
      #CONTROL Z #
ELSE IF X31 =%1 THEN #DIAL SYMBOL# 4
ELSE IF X31 =%( THEN #OPENING PARENTHESIS# 5
ELSE IF X31 =%) THEN #CLOSING PARENTHESIS# 6
ELSE IF X31 =%# THEN #COMMENT DETECTED# 7
ELSE IF X31 =%; THEN #END OF LIST OR STATEMENT# 8
ELSE IF X31 =%: THEN #LABEL DECLARED# 10
ELSE IF (X31 =%=)! (X31 =%0) THEN #ASSIGN# 11
ELSE IF X31 =%% THEN #CHARACTER VALUE# 12
ELSE IF X31 =%" THEN # " " STRING# 13
ELSE IF (X31 =X32 )!(X31 =0)!(X31 =B211)!(
      (X31 =B212)!(X31 =B200)!(X31 =LN)
      !(X31 =B214)
THEN #SPACE,TAB,LINE FEED,CARR.RETURN,CNTRL L# 14
ELSE 20
      ))))))))FI )));

EXPR(G21 : #PRINT ONE CHARACTER# P 3:=X31 );

EXPR(G11 : #PRINT,READ,CLASSIFY# VG21 ;VG4 );

EXPR(G9 : #PUSH LEVEL# X1 :=B5000+(X16 +:=1);
      X2 := X14 ; #STORE ELIFCOUNT# V G3 ;
      VG18 ; X14 :=0 );

EXPR(G10 : #POP LEVEL#
      V G12 ; #TEST COMMAFLAG#
      #DELETE FROM LIST# X20 :=X19 ;
      WHILE X14 >0 DO( X14 -:=1;V G13 );
      WHILE X20 < X18 DO(
            IF ((X27 :=C X20 )=B6000+X16 )!
            (X27 =B6000+X16 +1) THEN
            C X20 :=-1;X15 :=C(X20 +1);
            X17 :=C(X20 +2) #RESTORE #
            ELSE IF X27 =B0777= X16 THEN
            X27 -:=X16 ;
            IF X27 =B4000 THEN X31 :=%
            ELSE IF X27 =B5000 THEN X14 :=
            C(X20 +1)
            )FI; C X20 :=-1
            )FI ;
            X20 +:=5 #NEXTCHR NAME#
      );
      X16 -:=1;
      X21 :=X20 := X19 #RESET LIST POINTERS# );

EXPR(G13 : P 3:=%) #OUTPUT A ) SYMBOL#);

EXPR(G18 : #SET NAME ON LIST# V G15 #FLGHLF:=TRUE#
      WHILE X28 DO(
            IF X21 < X18 THEN
            IF C X21 =-1 THEN V
            G22 ELSE X21 +:=5
            FI
            ELSE X18 +:=5;V G22 #COPY AFTER LIST#
            FI
      )
);

```




```

EXPR(G22 : #COPY NAME FROM BUFFER TO LIST CELL#
      X33 :=A X1 -1;X27 :=X21 -1;
      WHILE (X33 +:=1)<A X21 DO(
        C (X27 +:=1):=C X33 ) ;
      V G16 #FLGHLP:='FALSE#);

```

```

EXPR(G7 : #SKIP COMMENT# V G11 ;WHILE X25 /=7
      DO( V G11 ) ; V G11 );

```

```

EXPR(G14 : #NAME DETECTED;SETUP BUFFER#
      X27 :=X1 :=X2 :=X3 :=X4 :=X5 :=
      X6 :=X7 :=X8 :=X9 :=X10 :=X11 :=X12 :=0 ;
      V G15 #FLGHLP:='TRUE# ;
      WHILE (X28 &(X27 <6)) DO(
        IF (X25 =1)|(X25 =2) THEN
          BEGIN C (A X7 +X27 ):= X31 ;
            X31 -:=B0257;#BUILD NAME#
            C (A X2 +(X27 /2))+:=(IF (X27 -((X27 /2)*2))=0
              THEN X31 06 ELSE X31 FI);
            VG4 ;#READ AND CLASSIFY#
            X6 :=(X27 +:=1)
          END
          ELSE V G16 #FLGHLP:='FALSE#
            FI
        ) ;
      WHILE X28 &((X25 =1)|(X25 =2)) DO(
        V G4 #NAME LONGER THAN 6 CHAR'ISH ) );

```

```

EXPR(G15 :X28 :=TRUE);

```

```

EXPR(G16 :X28 :=FALSE);

```

```

EXPR(G23 : #L LABEL# X1 := X17 );

```

```

EXPR(G24 : #Y IDENTIFIER# X1 :=X17 +B1000 );

```

```

EXPR(G20 : #SPECIAL: AT LEVEL=POP THEN ) TO 1#
      V G9 ;X1 :=B4000+X16 ;V G18 );

```

```

EXPR(G25 : #LOOKUP IN TABLE#
      X20 :=X19 -5;X5 :=0;X13 :=TRUE;
      WHILE (X13 &((X20 +:=5)<X18 )) DO(
        X27 :=-1;V G15 #FLGHLP:='TRUE#;
        WHILE (X28 &((X27 +:=1)<4)) DO(
          IF C(A X1 +X27 )/=C(X20 +X27 ) THEN
            V G16 #FLGHLP:='FALSE# FI
          ) ;
          IF X28 THEN X13 :=FALSE ;X5 :=C (X20 +4)
          FI #NOT FOUND: LOOP#
        ) ;
      );

```

```

EXPR(G26 : #SEARCH X=NAME# X1 :=B3000;VG25 ;
      IF X5 =0 THEN #A NEW NAME#
      V ( EXPR (G27 :X1 :=B3000 ; X5 :=(X24 +:=1));
        V G18 #ADD TO LIST# ) )
      FI; XX #OUTPUT X# );

```

```

EXPR(G28 : #SEARCH G#
      X1 :=B2000;#SETUP G# V G25 #AND SEARCH# );

```

```

EXPR(G29 : #SEARCH LOOP UNTIL LEVEL 0#
      X1 +:=1; X5 :=0 ;
      WHILE (((X1 -:=1)&B0777)/=0) & (X5 =0) DO(

```



```

EXPR(G36 : #OUTPUT IDENTAR# P 3*=(DECS(X5 )))." "
))

EXPR(G12 : #TEST ; ) COMBINATION/OUTPUT "SKIP" IF SO#
IF X26 THEN P 3*="SKIP "IV G3 #RESET# FI ;

EXPR(G5 : #IDENTIFIER DETECTED# VG14 ;#READ NAME#
IF X25 =7 #COMMENT# THEN V G7 FI;
V G8 ; #SKIP SPACES,LINE FEEDS ETC#
IF X25 =10 THEN #1 DETECTED#
BEGIN
    VG4 ; #READ NEXTCHR#
    P 3:= IF X25 =11 #1=# THEN
    BEGIN
        V G30 ; #SEARCH Y#
        IF X5 =0 THEN V G26 #X IDENTIFIER#
        ELSE XY#Y IDENTIFIER#
        FI
    END
    ELSE #LABEL#
        VG31 ; #SEARCH L#
        IF X5 =0 THEN #G# VG32 ; %G
        ELSE XL
        FI
    FI ;
    VG36 ; #OUTPUT CHAIN NUMBER# P 3:=%:
END
ELSE
    IF X25 =5 #X( # THEN #PROCEDURE NAME#
    BEGIN
        VG32 ; #SEARCH AND DEFINE G#
        VG20 ; #SUB CHAR. TO LIST#
        P 3*="SIG";VG36 ;X31 :=%,IV
        G11 #OUTPUT SIG.....#
    END
    ELSE
    P3:= BEGIN
        VG31 ; #LOCAL LABEL?#
        IF X5 =0 THEN
            IF X30 THEN VG32 ; %G #FUNCTION#
            ELSE
                BEGIN V G30 ; #SEARCH Y#
                IF X5 =0 THEN
                    BEGIN V G28 ;#SEARCH G-LABEL#
                    IF X5 =0 THEN V
                    G26 ELSE %G
                    FI
                END
                ELSE %Y
                FI
            END
            FI
        ELSE
            %L
            FI
        END ;
        V G36 #PRINT NUMBER#
    FI
    FI ;V G3 #RESET FLGCONMA AND FLAGV# ))

SUBR(G19 : #LIST OF NAMES# DCL(1); X29 :=TRUE;
WHILE X29 DO(

```



```

IF X25 =8 THEN X29 :=FALSE  #; IS LIST END#
ELSE IF X25 =1 THEN

```

```

BEGIN

```

```

V G14 ; #READ NAME#

```

```

CASE Y1 IN

```

```

#Y# V G34 ,

```

```

#L# V G35 ,

```

```

#G# V G32 ,

```

```

#X# V G27 , #VECTOR LIST#

```

```

#X# BEGIN VG27 ; X27 :=0 ;#ARRAY#

```

```

WHILE X25 /=2 DO( V G4 ) ;

```

```

#READ UNTIL A NUMERICAL CHARACTER#

```

```

WHILE X25 =2 DO(

```

```

X27 :=(X27 *10)+(X31 -X0) ;

```

```

V

```

```

G4 ) ; X24 +=

```

```

X27 END

```

```

ESAC

```

```

END

```

```

ELSE V

```

```

G4 )FI

```

```

) ; V

```

```

G4 ) ;

```

```

TABLE(G1 : 2,XI,XF,B0001,

```

```

2,XD,XO,B0002,

```

```

4,XC,XA,XS,XE,B0003,

```

```

5,XB,XE,XG,XI,XN,B0004,

```

```

2,XF,XI,B0011,

```

```

2,XO,XD,B0012,

```

```

4,XE,XS,XA,XC,B0013,

```

```

3,XE,XN,XD,B0014,

```

```

6,XV,XE,XC,XT,XO,XR,B0021,

```

```

5,XA,XR,XR,XA,XV,B0022,

```

```

6,XG,XL,XC,XB,XA,XL,B0041,

```

```

5,XL,XA,XB,XE,XL,B0042,

```

```

6,XF,XO,XR,XN,XA,XL,B0100,

```

```

5,XL,XO,XC,XA,XL,B0100,

```

```

3,XV,XE,XC,B0201,

```

```

1,XV,B0400,

```

```

4,XE,XL,XI,XF,B1000 ,

```

```

4,XS,XU,XB,XR,B2000,

```

```

G2 :4,XE,XX,XP,XR,B2000 )

```

```

END;

```

```

SKIP ) ;

```



6.10 PROPOSALS FOR A NEW MIDIAL PREPROCESSOR

If the preprocessor is bootstrapped to another computer with a bigger program store, the following restrictions of the preprocessor can be alleviated:

1. Allow comments in the lists of the declarational operators.

The way how this can be done is given in the flow diagram of the LIST subroutine of the preprocessor. (see page 79)

Still another check on comment must be made then within the array part of this subroutine at the reading of the number of array elements.

2. The locality of X and G identifiers within the body of a PROG is not incorporated in the current preprocessor.

This can be corrected so that a local list in the preprocessor elaboration the secondary of PROG of the X and G identifiers is built up.

The level part of a word of the identifier list can be increased by B100 for every new entered PROG.

A new save word for all the counters can be created as class number 7.

3. The searching of identifiers or of save words , which is now searching through the whole list, can be speeded up by linking all L-labels, all Y-identifiers, all special words etc. to each other as an L-string, Y-string and a save word string, respectively.

6.11 EXAMPLES.

THE DAY OF THE DATE

The program must find the day of a typed-in date.

The formula for the Gregorian day is :

$$r := (\text{day} + (\text{month} + 1) * 13 \div 5 + \text{year} + \text{year} \div 4 + \text{century} \div 4 - 2 * \text{century}) \bmod 7 ,$$

where $r=0$ gives SATURDAY as day.

January must be computed as the 13-th month, and February as the 14-th month, of the preceding year .

```
( *PROGRAM THAT CALCULATES THE DAY OF THE DATE,
  H.LUIIJF 75/01/17
```

```
  #
  !SUBR(MOD: !FORMAL X,Y ; X=(X/Y*Y) ) ;
  !SUBR(DATIN: !FORMAL N;!LOCAL X; #READ N DIGITS AS NUMBER#
    X:=0; NA(X:=(!DIGIT 1-X*10)+(Y*10));X );
  !P1*="DAY OF DATE.0YYYYY/MM/DD0";
```

```
!WHILE !TRUE !DO
```

```
  YEAR:=DATIN(4); !P1:=X/ ;
```

```
  !IF (( MONTH:=DATIN(2) )=1 )!(MONTH=2) !THEN MONTH+:=12;YEAR-:=1 !FI
```

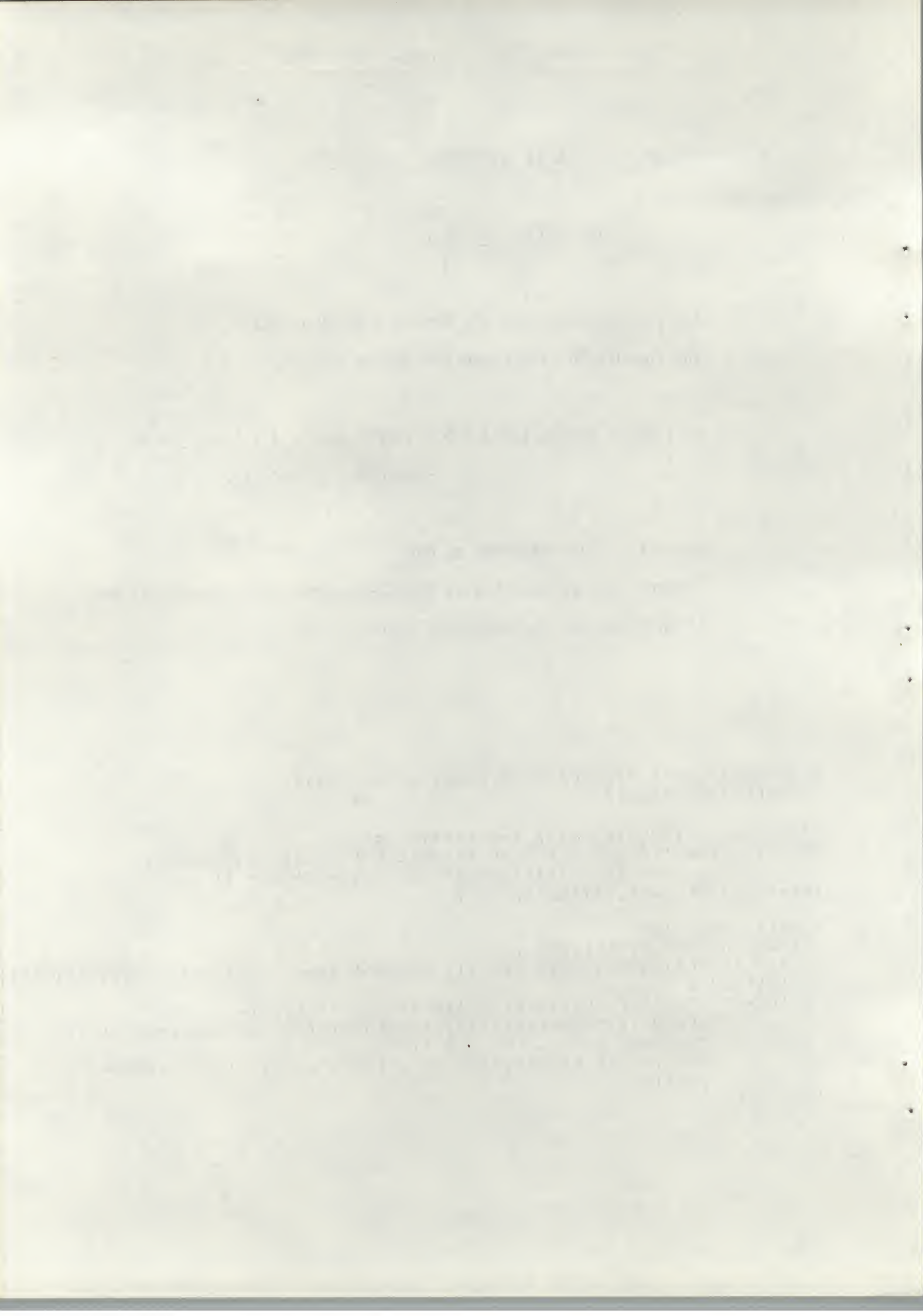
```
  !P1:=X/ ;
```

```
  !P1*=" ",!CASE (DATIN(2) #DAY# +(MONTH+1*13/5)+
```

```
    ((CENTURY:=YEAR/100)/4)-(2*CENTURY)+(YEAR:=YEAR..MOD..100)
    +(YEAR/4) +77 )..FCO.. 7 !IN
```

```
    "SUN","MON","TUES","WEDNES","THURS","FRI" !OUT "SATUR" !ESAC
    ."DAY0"
```

```
!ODD ) ;
```



The Post algorithm. (4)

Given a finite string S of 0's and 1's.

If the first digit of the string is a 1, append 1101 to the string and delete the first three digits. If the first digit is a 0, append 00 to the string and delete the first three digits.

As you can see, if there is a 0 as the first digit then the string becomes shorter, if it is a 1 the string will grow.

If it is not yet known whether there are strings that will grow forever.

There are strings which end in 00 (111 111 111 111 111 is a very good example) or strings which become periodic.

The following Midial program allows you to try a self chosen string to be treated by this algorithm. Type the string; every character which is not 0 or 1 starts the algorithm.

Notice the use of the SL (slice) operator in combination with the use of the vector length.

```

      *POST ALGORITHM      ,H.LUIJF      75/01/16#
(*WHILE 'TRUE 'DO      RCNT:=0 ;
  'P1:="00POST ALGORITHM, TYPE STRING ...0" ;
  'VECP:='VEC() ;

      'WHILE ((NUMBER:='R1')=X0)!(NUMBER=X1)
      'DO 'VECP:='VEC(NUMBER)      *READ STRING# 'DO ;
  'P1:='LN ; RCNT:=0 ;

  'WHILE 'C(VECP)>2 'DO
    'IF 'C(VECP+1)=X0 'THEN
      'VECP:=(VECP 'SL 'VEC(4,'C VECP )) . "00"
    'ELSE
      'VECP:=(VECP 'SL 'VEC(4,'C VECP))."1101"
    'FI ;
    CNT:=VECP;
    'WHILE (CNT+:=1)/=(VECP+'C VECP +1) 'DO 'P1:='C CNT 'DO ;
    'P1:='LN ;
    'IF (RCNT+:=1)=4 'THEN RCNT:=0 ; 'RESET 'FI
  'DO
'DO      );

```

THE UNIVERSITY OF CHICAGO

DEPARTMENT OF CHEMISTRY

RECEIVED FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO

ON THE 10th DAY OF JANUARY 1961

BY THE LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

LIBRARY OF THE UNIVERSITY OF CHICAGO

In the first part of the program the initial string is loaded and assembled by means of the concatenation.

In the second part the algorithm is programmed, the vector is sliced from the fourth element till the last one, and either "00" or "1101" is concatenated.

Because of the fact that the vectors are created in the vector area, we can not print with the vector assignment operator. Therefore the printing is done element by element. The vector stack is reset when RCNT becomes the value of 4.

This program can be shortened :

The algorithm for creating the next string can be compressed when the vector is sliced first and then the conditional vector concatenation is done by means of a nonvoiding IF clause.

The final program becomes now :

```

      $POST ALGORITHM ,H.LUIIJF 75/01/16#
('WHILE 'TRUE 'DO RCNT:=0 ;
  'P1*="00POST ALGORITHM, TYPE STRING ...0" ;
  'VECP:='VEC() ;

      'WHILE ((NUMBER:='R1)=%0)!(NUMBER=%1)
      'DO 'VECP,:='VEC(NUMBER) #READ STRING# '00 ;
      'P1:='LN ; RCNT:=0 ;

      'WHILE 'C(VECP)>2 'DO
        'VECP:=(VECP 'SL 'VEC(4,'C VECP ))
        'IF 'C(VECP+1)=%0 'THEN "00" 'ELSE "1101" 'FI ;
      'P1*='VECP,"0"
      'DO
      '00 )

```

It is a very old and famous town, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place. It is a very old and famous town, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place.

The town is very old and famous, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place. It is a very old and famous town, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place.

The town is very old and famous, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place. It is a very old and famous town, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place.

The town is very old and famous, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place. It is a very old and famous town, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place.

The town is very old and famous, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place. It is a very old and famous town, and has been so for many centuries. It is situated on a hill, and is surrounded by a wall. The town is very beautiful, and is a very important place.

A PROGRAM EXECUTIVE.

This program is a program which encloses four PROG's (programs) , and one program which can be loaded into a core buffer from a batch string of program binaries from the highspeed input channel.

Inspection of the program, reveals that a vector of program address pointers is created. With the declaration operator VECTOR the identifiers of the main program are located in consecutive memory locations.

Thereafter three memory instructions of the loader are modified with no-operation instructions (B7000, see page 102) .

Four programs are defined. They reside permanently in core. First the program number 5 is called with the CALLPR (-ogram) operator. Typing in a digit between 2 and 5 executes one of the incorporated programs (list of squares, plot of $Y^2/4$, the triangle of Pascal and the list of possibilities.) .

If the digit 0 is typed in, the maximum number of X-store locations available is declared, this number is placed in the location of the load buffer begin. (LOAD) Thereafter the loader is called, the loaded program address pointer (program 1) is placed on the location following the load buffer begin, after the program vectorlength.

The X-store is shortened to the number of X-store locations used within the program executive added to the number of X-store locations used by the loaded program. (without its own X-store, this is declared at the program call with CALLPR.)

Typing in another digit greater than 5, the executive execution is ended and the control returns to the Monitor.

Note that the typed-in digit is read with DIGIT and delivers the character value of the typed digit.

THE HISTORY OF THE
CITY OF BOSTON

From its first settlement in 1630 to the present time, the city of Boston has been a center of commerce and industry. It has been the seat of government, the home of the courts, and the center of education. It has been the birthplace of many of the great men of the country, and the scene of many of the great events of our history. It has been the home of the Puritans, the Quakers, the Unitarians, and the Catholics. It has been the home of the Pilgrims, the Revolutionaries, and the Abolitionists. It has been the home of the great men of the country, and the scene of many of the great events of our history. It has been the home of the Puritans, the Quakers, the Unitarians, and the Catholics. It has been the home of the Pilgrims, the Revolutionaries, and the Abolitionists. It has been the home of the great men of the country, and the scene of many of the great events of our history.


```
#PROGRAM EXECUTIVE ,      H.LUIIJF      75/01/27  #

( I VECTOR N, PROGRAM, PR1, PR2, PR3, PR4, PR5, LOAD)
  IC 1B7676:=IC 1B7677:=IC 1B7700:=1B7700 ;

PR2:='PROG(#X AND X^2#X:=0; IWHILE (X:=1)<11 IDO
      IP1*="0".IDEC(S(X))." ".IDEC(S(X*X)) IDO;IP1:='LN );

PR3:='PROG(#PLOT OF X^2 /4# IP1*=56^X#; X:=0;
      IWHILE X/=16 IDO
      IP1*="0* ". ((X:=1)*X/4^'SP)."+ " ) IDO;IP1:='LN );

PR4:='PROG(#TRIANGLE OF PASCAL# IARRAY P(15); P*='VECT(1); X:=70;
      IWHILE P/=15 IDO
      IP1*=(X-:=5)/2^'SP; IP1*='1ST @ IDECS @ IA P;
      IP1:='LN; P*=(IA P, IVEC(0))+@ (IVEC(0), IA P)
      IDO
      );

PR5:='PROG(IP1*="00=LOAD01=EXEC02=X , X^203=PLOT04=PASCAL05=PROGRAMS0");
ICALLPR PR5;

IWHILE (IP1*="0PROGRAM NO." ; (N:=IDIGIT 1 -X0)<=5) IDO
  IF N=0 ITHEN
    IDCL(LOAD:=IC 1B0024-1AX0-100);
    S IVEC(1B7600,4, IA LOAD);PR1:='A LOAD+1;
    IDCL(IA LOAD-1AX0+LOAD);
    IP1*=" LOADED"
  ELSE
    IP1:='LN ; ICALLPR IC (IA PROGRAM+N)
  IFI
IDO
))
```

```
#THE TRIANGLE OF PASCAL #
IBEGIN IARRAY PASCAL(15);
#RESERVE VECTOR ROOM#

LNCNT:=70 ; #LENGTH OF OUTPUT LINE#
PASCAL*='VECT(1) ; #VECTOR INITIALISATION#
IWHILE PASCAL /= 15 #VECTOR LENGTH AS COUNT#
IDO
  IP1*=(LNCNT-:=5)/2 ^'SP ; #CREATE THE LEADING BLANKS#
  IP1*=(1ST @ IDECS @ IA PASCAL ), IVEC('LN) ;
  #OUTPUT THE STRAIGHTENED NUMBER VECTOR#

#CALCULATE NOW THE NEXT LINE :
  VECTOR LENGTH FOLLOWED BY THE NUMBERS
  CONCATENATE WITH ZERO AND ADD THE INVERSE VECTOR.
  THE SECOND VECTOR : 2.1.1 CONCATENATED BECOMES :
  3.1.1.0 AND MAKE THE INVERSE VECTOR :
  3.0.1.1 ADDED BECOMES THIS
  3.1.2.1 ETCETERA.....#
PASCAL*=(IA PASCAL, IVEC(0))+@ (IVEC(0), IA PASCAL)
IDO
))
```

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

PROBLEM: THE TOWERS OF HANOI. (4)

On a board there are three pegs. On the first peg there is initially a stack of disks. These disks are not equal of diameter.

The disk on the top has the smallest diameter, and going to the stack bottom, the diameter of each successive disk is greater than the diameter of the preceding one.

The problem is how to move all disks, one by one, to the second stack with the restriction that never a larger disk shall be above a smaller disk.

The third stack may be used for shunting.

If you have one disk, the problem is solved by transferring the disk from stack 1 to stack 2.

If you have m disks, you solve the problem by moving $m-1$ disks to the shunt stack (3), moving thereafter the m -th disk to stack 2 and moving back the $m-1$ disks from peg 3 to the peg 2.

It can be proved that $2^m - 1$ movements are needed for solving the problem for m disks.

```
# TOWERS OF HANOI ,      75/01/14  #

( 'SUBP(HANOI: 'FORMAL X,Y,Z,N ;
  'IF N>0 'THEN HANOI(X,Z,Y,N-1);
  'P1*="01 DISK FROM ".X." TO ".Y ;
  HANOI(Z,Y,X,N-1) 'FI );

# TEXT ASSIGNATION IN THE X-STORE ; RESERVE POOL #
'ARRAY TK1(7),TK2(7),TK3(7) ;

TK1*="STACK 1";TK2*="STACK 2";TK3*="STACK 3";
'WHILE 'TRUE 'DO
  'P1*="00NUMBER:" ;
  NUMBER:='DECIN(1) ;
  'P1*="0TOWERS OF HANOI FOR: ".DECN(NUMBER)." DISKS0";
  HANOI('A TK1,'A TK2,'A TK3,NUMBER) 'DO ) ;
```

THE HISTORY OF THE

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...
... of the ...

... of the ...
... of the ...
... of the ...

NUMBER: 1
TOWERS OF HANOI FOR: 1 DISKS

1 DISK FROM STACK 1 TO STACK 2

NUMBER: 2
TOWERS OF HANOI FOR: 2 DISKS

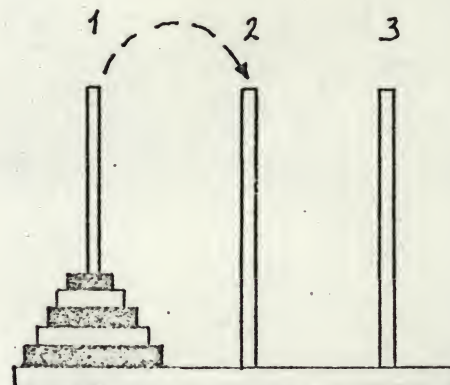
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 2

NUMBER: 3
TOWERS OF HANOI FOR: 3 DISKS

1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 2

NUMBER: 4
TOWERS OF HANOI FOR: 4 DISKS

1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 2 TO STACK 1
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 2 TO STACK 1
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 2



NUMBER: 5
TOWERS OF HANOI FOR: 5 DISKS

1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 2 TO STACK 1
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 2 TO STACK 1
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 1 TO STACK 3
1 DISK FROM STACK 2 TO STACK 3
1 DISK FROM STACK 1 TO STACK 2
1 DISK FROM STACK 3 TO STACK 1
1 DISK FROM STACK 3 TO STACK 2
1 DISK FROM STACK 1 TO STACK 2

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100



101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300

7. REFERENCES.

- (1) MINIALGOL 8.
- P.G. HIBBARD
Dept. of Computational and Statistical Science
University of Liverpool.
- (2) FLOWCHART TECHNIQUES FOR STRUCTURED PROGRAMMING.
- I.Nassi and B.Shneiderman
SIGNPLAN Notices , august 1973 ,pg. 12-26.
- (3) REPORT ON THE ALGORITHMIC LANGUAGE ALGOL 68.
- A.van Wijngaarden, B.J. Mailloux ,
J.E.L. Peck and C.H.A. Koster.
NUMERISCHE MATHEMATIC 1969, February.
- (4) THE PROGRAMMING LANGUAGES LISP and TRAC.
- Prof.dr.ir. W.L. van der Poel
Collegedictaat T.H. Delft, 3^e edition, 1972.
- (5) INTRODUCTION TO PROGRAMMING.
PDP 8/E SMALL COMPUTER HANDBOOK.
- (6) OS/8 HANDBOOK.
PS/8 PROGRAMMING SYSTEM.
- all by Digital Equipment Corporation.
- (7) COMPUTATION: FINITE AND INFINITE MACHINES.
- M.L. Minsky
Prentice-Hall, Englewood Cliffs ,N.J.; 1967.
- (8) PDP 8 COOKBOOK.
- F. Anthoni , Medisch-Biologisch Laboratorium
RVO/TNO , Rijswijk .

1. Introduction

2. Methodology

3. Results and Discussion

4. Conclusion

5. Acknowledgements

6. References

7. Appendix

8. Glossary

9. Index

10. Bibliography

11. List of Figures

12. List of Tables

13. Summary

14. Abstract

15. Executive Summary

16. Introduction

17. Methodology

18. Results and Discussion

19. Conclusion

20. Acknowledgements

21. References

22. Appendix

23. Glossary

24. Index

25. Bibliography

26. List of Figures

8.1 LIST OF ERROR CODES.

I. In the MIDIAL system.

1. "SYS ERR AT nnnn" -codes :

with .nnnn=

- | | |
|------|--|
| 0000 | Jump to location 0 or 1.
This could be an illegal indirect
subroutine entry in the minimised system. |
| 0316 | Error in the divide subroutine. |
| 7617 | Input for the loader is not a MINIAL
binary. (starts not with <u>B</u> 125) |
| 7632 | Program binary too long for the reserved
program core . |
| 7670 | Recursive call of the loader in error. |
| 7706 | Checksum of the loader input and the
computed checksum not equal. |

2. "I/O ERR AT nnnn" - codes :

- | | |
|------|--|
| 4660 | Try to read from the R3/R4 input device
when there is no device or file specified
as bufered input . |
| 5010 | Writing a buffer to the output device
in error . |
| 5017 | Output device full . |
| 5046 | Fetching of the input handler in error . |
| 5241 | Reading a buffer from an input device
in error. |



- 5254 The end of the input file is reached while trying to read more input data.
- 5323 Fetching the output handler in error.
- 5342 Closing error of the output file.
- 5450 Error in entering the output file.
- 5467 Directory device on output acts as a non-directory device at entering the output file.
- 5540 Error in the lookup of a file (-name).

3. "n) MISSING" - code :

This is a warning error of the preprocessor. In the treated program there are n close tokens to less. They are inserted automatically at the program end.

II. In the MIDICM system :

1. "SYS ERR AT nnnn" -codes :

- 0000 Jump to location 0 or 1.
This error may not appear in this system.
- 0324 Divide error.
- 0676 Recursive call of a protected non-recursive subroutine. (SETFLAG error)
- 5476 Compiler error : generated after ERR 9 or label list full .
- 7617 Input for the loader is not a MINIAL binary. (it starts not with B125 as leader frame)

EXPR(G30 : #SEARCH Y IDENTIFIER# VG24 ; #SETUP# V G29 #SEARCH#);

EXPR(G31 : #SEARCH L=LABEL# VG23 / #SETUP L# V G29);

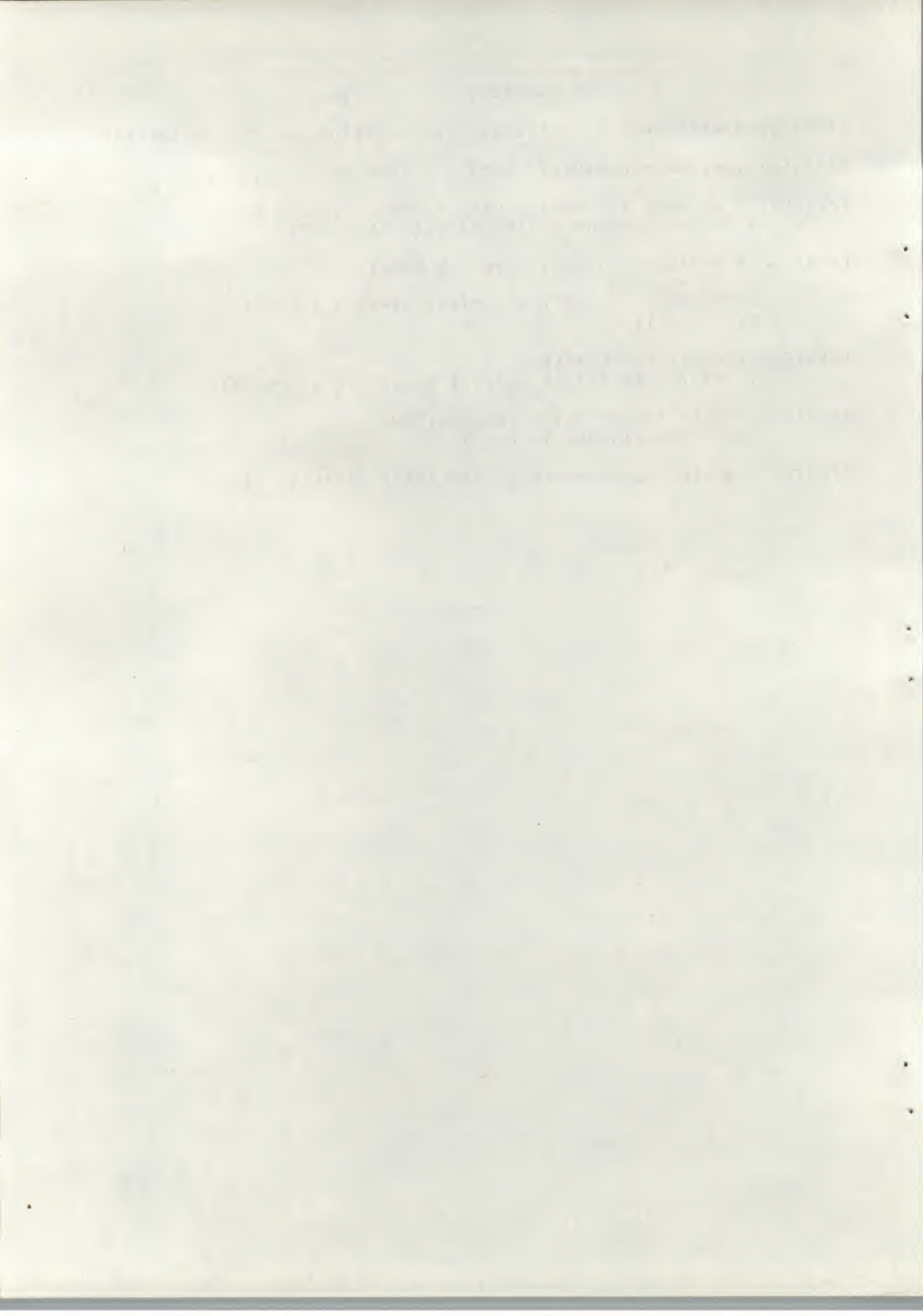
EXPR(G32 : #SEARCH AND DEFINE G# V G28 ; #SEARCH G#
IF X5 =0 THEN X5 :=(X23 +=1); V G18 FI);

EXPR(G33 : #SEARCH AND DEFINE Y# V G30 ;
IF X5 =0 THEN
VEXPR (G34 : V G24 ; X5 :=(X15 +=1) ; V G18)
FI);

EXPR(G35 : #DEFINE L=LABEL#
V G23 / X5 :=(X22 +=1) ; V G18 #TO LIST#);

EXPR(G8 : #SKIP AND OUTPUT CR,LF,TAB,SP#
WHILE X25 =14 DO(V G11));

EXPR(G3 : #RESET FLGCOMMA FLAG# X26 :=X30 :=FALSE);



- 7632 Program binary too long for the reserved program core.
- 7670 Error at the recursive call of the loader.
- 7706 Checksum error : the computed checksum of the loader input does not match with the checksum frames of the input.

2. "I/O ERR AT nnnn" - codes:

- 4660 Try to read from the R3/R4 input device when there is no input device and/or file specified in the file command string.
- 4725 Writing an output buffer to the output device in error.
- 4734 Output device full. (the file is closed!)
- 5056 Fetching the input handler in error.
- 5241 Reading a buffer from the input device in error.
- 5254 The end of the input file is reaching while trying to read more input data.
- 5323 Fetching the output handler in error.
- 5346 Closing error at the output file.
- 5450 Error in entering the output file.
- 5467 The directory device on output acts as a non-directory device at entering the output file (device full)
- 5522 The directory device on input acts as a non-directory device at entering the input file.
- 5540 Error in the lookup of a file (-name).

3. "ERR n" -codes (program errors)

- ERR 1 A character sequence does not form a basic symbol in the language MINIAL, e.g. ISAC for ESAC .
Action : The invalid symbols are ignored.
- ERR 2 A syntax error has occurred. This can occur when a basic symbol in MIDIAL was not preceded by a ' symbol, or when a ;-symbol is missing.
Action : the compiler skips up to the next semicolon and continues to compile from there.
- ERR 3 A void quaternary has been used in a nonvoid context. This can occur if a semicolon preceeds a close token in the program. A SKIP operator is inserted there by the preprocessor.
- ERR 4 A dyadic operator has been used in a monadic context, e.g. DIV := /8 ;
- ERR 5 An illegal operator and becomes is used, e.g. COUNT+*= 7 or a †:= .
- ERR 6 A label has more than one defining occurrence in the same procedured secondary.
- ERR 7 A label has an applied occurrence, but no defining occurrence.
- ERR 8 An illegal secondary has been used on the left-hand side of an assignment operator, e.g. CNT-3:= 9 ;
- ERR 9 The program is too large to be compiled with the current compiler and the current system.
This will almost invariable mean that too many labels have been used. After this error the system generates : "SYS ERR AT 5476" .

After ERR 3 till 9 checking continues but the code generation is in error.

8.2 LIST OF IMPORTANT MACHINE ADDRESSES.

- B 0000 - Subroutine entry of the HALT operator and of the fault routines. On this address the program counter (call address plus one) is set.
- B 0001 - First executional instruction of the error routine.
- B 0010 - This address contains the stackpointer (STP).
- B 0023 - This address contains the vector area top pointer (VAP).
- B 0024 - This address contains the vector bottom pointer (VAPB).
These two addresses can be used for providing the destruction of the vector stack by a vector assignation:
.....; VAPB:=C B0024; C B0024:=C B0023;..... *= ...
#critical region# ; C B0024:=VAPB ;.....
But it is preferred to procedure this critical region :
.....; \$ SUBR(.. *= ... #critical region#) ;.....
- B 0025 - This address contains the address AW 0 (begin of the common variable store or W-store) .
- B 0026 - This address contains the address AX 0 (begin of the variable store or X-store) .
This value is also the address of the program store end.
The loaded program can be printed out as follows :
(LABEL LPRLDTRACE ; PRCNT:=LPRLDTRACE-1 ;
WHILE (PRCNT+=1)< AX 0 DO
P1 *= OCTS(PRCNT) ." ".OCTS(C PRCNT).VEC(LN)
OD ; LPRLDTRACE : #program to trace#
.....);



- B 0027 - This address contains the address AY 0 .(begin of the parameter and local variable store)
- AX 0 - This address contains initially the number of declared X-store locations of the current program.
This number is changed if a use is made of X 0 to store data.
- AY 0 - This address contains initially the number of transferred parameters of a called subroutine.
This number can be changed by declaring parameters with the DCL operator or with the LOCAL or FORMAL operator. It is also changed by an assignation to Y 0.
- B 7600 - This is the start address of the system loader.

By means of these addresses a dump routine can be made of the system addresses :

```
SUBR(DUMP: P1*="\DUMP VAP:".OCTS(CB0024))."\VAPB:".OCTS
(C Y -2)."\AX0:".OCTS(CB0026))."\AY0:".
OCTS(C Y -1) ; LOCAL X ; P1*="\STACK:\"
(( C Y -2)-(X:=CB0024))+(P1*="\OCTS(C(X+:1)) ) ;
```

Notice the use of the just VAPB and AY0 pointers by taking them of the old environment. (CY-2 and CY-1).

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

THE HISTORY OF THE

8.3 The entry table of the subroutines.

The subroutines of the Midialgol system can be used by the user by means of the code statements. The octal value in the octal integer list must be equal to B4400 plus the table entry number, except if the entry is marked with %.

<u>table entry number</u>	<u>octal entry</u>	<u>subroutine</u>
0000 ₈ %	4000	error return routine : (HALT) prints "SYS ERR AT" .
0030	4430	push accu; load 1 ;
0031	4431	push accu; load 2 ;
0032	4432	push accu; load 3 ;
0033	4433	push accu; load 4 ;
0034	4434	push accu; load 5 ;
0035	4435	push accu; load 6 ;
0036	4436	push accu; load 7 ;
0037	4437	push accu; load 10 ₈ ;
0040	4440	push accu; load 11 ₈ ;
0041	4441	push accu; load 12 ₈ ;
0042	4442	RESET subroutine ; (VAP:=VAPB)
0043	4443	monadic user defined operator call: .. label identifier .. .
0044	4444	monadic minus operator.
0045	4445	AP or load TRUE .
0046	4446	CALLPR operator subroutine.
0047	4447	M operator subroutine.
0050	4450	N operator subroutine.
0051	4451	V operator subroutine.
0052	4452	\$ operator.
0053	4453	X operator: accu:=C(AX0+accu)
0054	4454	Y operator: accu:=C(AY0+accu)

<u>table entry number</u>	<u>octal entry</u>	<u>subroutine</u>
0055	4455	Z operator: $\text{accu} := \text{CC}(\text{AYO} + \text{accu})$
0056	4456	W operator: $\text{accu} := \text{C}(\text{AWO} + \text{accu})$
0057	4457	R operator. (read)
0060	4460	C operator. (contents of)
0061	4461	COPY operator.
0062	4462	ST operator. (straighten)
0063	4463	DECIN operator. (read number)
0064	4464	OCTIN operator. (read octal integer)
0065	4465	DECS operator. (output number)
0066	4466	OCTS operator. (output octal integer)
0067	4467	CHAR subroutine.
0070	4470	ALPHA operator subroutine.
0071	4471	DIGIT operator subroutine.
0072	4472	@ operator .
0073 %	5473	GOTO the contents of the accu: GOTOAC, and POP the stack.
0074	4474	DCL operator.
0075	4475	assign.
0076	4476	assign and pop the stack. (void)
0077	4477	vector assign.
0100	4500	vector assign and pop the stack. (voiding vector assignment)
0101	4501	assign and becomes subroutine.
0102	4502	user defined dyadic operator call: .. label identifier ...
0103	4503	creation operator.
0104	4504	free subroutine: the old ' operator of Minialgol.
0105	4505	& (and) operator subroutine.

<u>table entry number</u>	<u>octal entry</u>	<u>subroutine</u>
0106	4506	! (or) operator.
0107	4507	+ (add) operator.
0110	4510	- (subtract) operator.
0111	4511	← (shift left) operator.
0112	4512	= (equal) operator.
0113	4513	/=(not equal) operator.
0114	4514	> (greater) operator.
0115	4515	>=(greater or equal) operator.
0116	4516	< (less then) operator.
0117	4517	<=(less then or equal) operator.
0120	4520	* (multiply) operator.
0121	4521	/ (divide) operator.
0122	4522	ADD operator entry of Minialgol: not used in this system. ADD is the double length dyadic add subroutine.
0123	4523	MULT operator: dyadic double length multiplication subroutine: not used entry of this system.
0124	4524	DIV operator: dyadic double length divide operator subroutine entry: not used in this system. These double length entries are shortened to the standard length routines. (ADD≡+ etc.).
0125	4525	. (concatenation) operator.
0126	4526	SL(slicing) operator.
0127	4527	MEM (member of) operator.
0130	4530	HD (head of) operator
0131	4531	COMP(compress) operator.
0132	4532	EXPAND operator.

Summary of the results of the experiments

Experiment 1	100	100
Experiment 2	100	100
Experiment 3	100	100
Experiment 4	100	100
Experiment 5	100	100
Experiment 6	100	100
Experiment 7	100	100
Experiment 8	100	100
Experiment 9	100	100
Experiment 10	100	100
Experiment 11	100	100
Experiment 12	100	100
Experiment 13	100	100
Experiment 14	100	100
Experiment 15	100	100
Experiment 16	100	100
Experiment 17	100	100
Experiment 18	100	100
Experiment 19	100	100
Experiment 20	100	100
Experiment 21	100	100
Experiment 22	100	100
Experiment 23	100	100
Experiment 24	100	100
Experiment 25	100	100
Experiment 26	100	100
Experiment 27	100	100
Experiment 28	100	100
Experiment 29	100	100
Experiment 30	100	100
Experiment 31	100	100
Experiment 32	100	100
Experiment 33	100	100
Experiment 34	100	100
Experiment 35	100	100
Experiment 36	100	100
Experiment 37	100	100
Experiment 38	100	100
Experiment 39	100	100
Experiment 40	100	100
Experiment 41	100	100
Experiment 42	100	100
Experiment 43	100	100
Experiment 44	100	100
Experiment 45	100	100
Experiment 46	100	100
Experiment 47	100	100
Experiment 48	100	100
Experiment 49	100	100
Experiment 50	100	100
Experiment 51	100	100
Experiment 52	100	100
Experiment 53	100	100
Experiment 54	100	100
Experiment 55	100	100
Experiment 56	100	100
Experiment 57	100	100
Experiment 58	100	100
Experiment 59	100	100
Experiment 60	100	100
Experiment 61	100	100
Experiment 62	100	100
Experiment 63	100	100
Experiment 64	100	100
Experiment 65	100	100
Experiment 66	100	100
Experiment 67	100	100
Experiment 68	100	100
Experiment 69	100	100
Experiment 70	100	100
Experiment 71	100	100
Experiment 72	100	100
Experiment 73	100	100
Experiment 74	100	100
Experiment 75	100	100
Experiment 76	100	100
Experiment 77	100	100
Experiment 78	100	100
Experiment 79	100	100
Experiment 80	100	100
Experiment 81	100	100
Experiment 82	100	100
Experiment 83	100	100
Experiment 84	100	100
Experiment 85	100	100
Experiment 86	100	100
Experiment 87	100	100
Experiment 88	100	100
Experiment 89	100	100
Experiment 90	100	100
Experiment 91	100	100
Experiment 92	100	100
Experiment 93	100	100
Experiment 94	100	100
Experiment 95	100	100
Experiment 96	100	100
Experiment 97	100	100
Experiment 98	100	100
Experiment 99	100	100
Experiment 100	100	100

<u>table entry number</u>		<u>octal entry</u>	<u>subroutine</u>
0133	%	5533	(jump to)the end of program routine.
0134	%	5534	(jump to)the end of subroutine subroutine.
0135	%	5535	(jump to)the end of expression routine.
0136	%	5536	(jump to)the end of called expressions (VEXPR's) subroutine.
0137		4537	push subroutine: load the argument while pushing the accu.
0140		4540	pop subroutine: <u>CB0174</u> (QLOC) :=accu; accu:=stack top contents.
0141		4541	GOTONV : goto the argument in the following location.
0142		4542	GOTOV : goto voiding=the users GOTO.
0143		4543	procedure routine: EXPR,SUBR and PROG cause this function. Evaluation of it causes a jump over the procedure body, while the accu is stacked and thereafter is loaded with the procedure call address
0144		4544	increment routine: increments the contents of the address in loaded in the accu.(or nonvoid value).
0145		4545	decrement routine (see increment).
0146		4546	open vector routine.
0147		4547	close vector routine.
0150		4550	string load routine: string follows in the next locations.
0151		4551	case in subroutine.
0152		4552	then handling routine.
0153		4553	push ; load AX (argument) ;

<u>table entry number</u>	<u>octal entry</u>	<u>subroutine</u>
0154	4554	push ; load AY{ argument);
0155	4555	push ; load X (argument);
0156	4556	push ; load Y (argument);
0157	4557	push ; load Z.(argument);
0160	4560	subroutine optimisation call routine.
0161	4561	enter subroutine: optimisation call.
0162	4562	SREG : push;load switch register ;
0163	4563	load the argument subroutine.
0164	4564	optimised expression call routine.
0165	4565	routine that adds its argument in the
		accu: accu+= argument ;
0166	4566	accu &:= argument ;
0167	4567	protection flag routine: protection of
		non recursive routines.
0170	4570	save: <u>C</u> <u>B0175</u> :=accu;
0171	4571	save: accu:= <u>C</u> (<u>C</u> <u>B0176</u> :=accu) ;

the following numbers can be added to the accu by coding B1000 plus the entry table number:

0172	%	1172	add -1
0173	%	1173	add -2
0012	%	1012	add 2
0013	%	1013	add 3
0014	%	1014	add 4
0015	%	1015	add 5
0016	%	1016	add 6
0017	%	1017	add 7
0020	%	1020	add 10 ₈
0021	%	1021	add 11 ₈
0022	%	1022	add 12 ₈

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

8.4 CHANGING THE P1 and P2 DEVICE IN A MIDIALGOL SYSTEM WITH INTERRUPT.

These changes have the form of $\underline{C}(\underline{C} \ \underline{B} \ 0075 + x) := y$; where x is an octal number. $\underline{C} \ \underline{B} \ 0075 + x$ is an address in the assignation routine which contains the machine instruction for the P1/P2 output sequence.

These are for x= B0031 - the character output load sequence.

B0032 - the character output flag test instruction.

B0037 - the interrupt on instruction.

The programmer can change with a Midialgol expression or subroutine the output device by changing the previous three instructions.

Note: Before every interrupt on instruction no output flags may be on .

For example : SUBR(CHANGE: FORMAL LOADSQ,TEST,INT ;

C(C B0075+B0031):=LOADSQ ;

C(C B0075+B0032):=TEST ;

C(C B0075+B0037):= INT);

EXPR(TTY: CHANGE(B6046,B6041,B6001);

EXPR(VISUAL: CHANGE(B6146,B6141,B7000); #INT:=NO-OPERATION#

EXPR(PTR: CHANGE(B6026,B6021,B7000);

EXPR(INTON: CODE(6001) # set the interrupt enable on#

EXPR(RESET: CODE(6042,6142,6022) ; V INTON);

reset all device flags

a program part could be as following:

V PTR ; P1:=%Q ; V RESET ; V TTY ; P1:=%T ;

after elaboration of this part a character Q is punched in papertape and the character T is transferred to the teletype.

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
CHICAGO, ILLINOIS

TO THE HONORABLE SENATE OF THE UNIVERSITY OF CHICAGO
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE DEPARTMENT OF CHEMISTRY
THAT THE THESIS OF
[Name] SUBMITTED BY [Name]
IN CANDIDACY FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE DEPARTMENT OF CHEMISTRY
BE ACCEPTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

IN TESTIMONY WHEREOF
THESE SIGNS AND SEALS
HAVE BEEN AFFIXED
THIS [Date] DAY OF [Month] 19[Year]

CHIEF OF DEPARTMENT
[Signature]
DEPARTMENT OF CHEMISTRY
UNIVERSITY OF CHICAGO
CHICAGO, ILLINOIS

DEAN OF THE UNIVERSITY
[Signature]
OFFICE OF THE DEAN
UNIVERSITY OF CHICAGO
CHICAGO, ILLINOIS

CHIEF OF DEPARTMENT
[Signature]
DEPARTMENT OF CHEMISTRY
UNIVERSITY OF CHICAGO
CHICAGO, ILLINOIS

8.5 THE ASCII CODE TABLE

last digit first two digits	ASCII code (octal 8-bits)							
	0	1	2	3	4	5	6	7
20	null	↑A	↑B	↑C	↑D	↑E	↑F	↑G/bell
21	↑H	hor. tab	line feed	vert. tab	form feed	carr. return	↑N	↑O
22	↑P	↑Q	↑R	↑S	↑T	↑U	↑V	↑W
23	↑X	↑Y	↑Z	alt. mode				
24	blank	!	"	#	\$	%	&	'
25	()	*	+	,	-	.	/
26	0	1	2	3	4	5	6	7
27	8	9	:	;	<	=	>	?
30	@	A	B	C	D	E	F	G
31	H	I	J	K	L	M	N	O
32	P	Q	R	S	T	U	V	W
33	X	Y	Z	[\]	↑	
37				(shift K)	(shift L)	(shift M)	(shift N)	rubout

